Part I provided a broad look at the fundamentals of all parts of networking, focusing on Ethernet LANs, WANs, and IP routing. Parts II and III now drill into depth about the details of Ethernet, which was introduced in Chapter 2, “Fundamentals of Ethernet LANs.”

Part II begins that journey by discussing the basics of building a small Ethernet LAN with Cisco Catalyst switches. The journey begins by showing how to access the user interface of a Cisco switch so that you can see evidence of what the switch is doing and to configure the switch to act in the ways you want it to act. At this point, you should start using whatever lab practice option you chose in the “Your Study Plan” section that preceded Chapter 1, “Introduction to TCP/IP Networking.” (And if you have not yet finalized your plan for how to practice your hands-on skills, now is the time.)

After you complete Chapter 4 and see how to get into the command-line interface (CLI) of a switch, the next three chapters step through some important foundations of how to implement LANs—foundations used by every company that builds LANs with Cisco gear. Chapter 5 takes a close look at Ethernet switching—that is, the logic used by a switch—and how to know what a particular switch is doing. Chapter 6 shows the ways to configure a switch for remote access with Telnet and Secure Shell (SSH), along with a variety of other useful commands that will help you when you work with any real lab gear, simulator, or any other practice tools. Chapter 7, the final chapter in Part II, shows how to configure and verify the operation of switch interfaces for several important features, including speed, duplex, and autonegotiation.

Part II

Implementing Ethernet LANs

CHAPTER 4

Using the Command-Line Interface

This chapter covers the following exam topics:

2.0 Network Access

2.8 Describe network device management access (Telnet, SSH, HTTP, HTTPS, console, TACACS+/RADIUS, and cloud managed)

The CCNA exam focuses on skills like understanding how LANs work, configuring different switch features, verifying that those features work correctly, and finding the root cause of the problem when a feature is not working correctly. Before doing the more important work, you must first learn how to access and use the user interface of the switch, called the **command-line interface (CLI)**.

This chapter begins that process by showing the basics of how to access the switch's CLI. These skills include how to access the CLI and how to issue verification commands to check on the status of the LAN. This chapter also includes the processes of how to configure the switch and how to save that configuration.

Note that this chapter focuses on processes that provide a foundation for the exam topics which include the verbs *identify*, *configure*, or *verify*. Most of the rest of the chapters in Parts II and III of this book then go on to include details of the particular commands you can use to verify and configure different switch features.

"Do I Know This Already?" Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book as well as on the companion website, includes both the answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 4-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Accessing the Cisco Catalyst Switch CLI	1–3
Configuring Cisco IOS Software	4–6

1. In what modes can you type the command **show mac address-table** and expect to get a response with MAC table entries? (Choose two answers.)

a. User mode

b. Enable mode

c. Global configuration mode

d. Interface configuration mode

2. In which of the following modes of the CLI could you type the command **reload** and expect the switch to reboot?
   - **a.** User mode
   - **b.** Enable mode
   - **c.** Global configuration mode
   - **d.** Interface configuration mode
3. Which of the following is a difference between Telnet and SSH as supported by a Cisco switch?
   - **a.** SSH encrypts the passwords used at login, but not other traffic; Telnet encrypts nothing.
   - **b.** SSH encrypts all data exchange, including login passwords; Telnet encrypts nothing.
   - **c.** Telnet is used from Microsoft operating systems, and SSH is used from UNIX and Linux operating systems.
   - **d.** Telnet encrypts only password exchanges; SSH encrypts all data exchanges.
4. What type of switch memory is used to store the configuration used by the switch when it is up and working?
   - **a.** RAM
   - **b.** ROM
   - **c.** Flash
   - **d.** NVRAM
   - **e.** Bubble
5. What command copies the configuration from RAM into NVRAM?
   - **a.** **copy running-config tftp**
   - **b.** **copy tftp running-config**
   - **c.** **copy running-config start-up-config**
   - **d.** **copy start-up-config running-config**
   - **e.** **copy startup-config running-config**
   - **f.** **copy running-config startup-config**
6. A switch user is currently in console line configuration mode. Which of the following would place the user in enable mode? (Choose two answers.)
   - **a.** Using the **exit** command once
   - **b.** Using the **end** command once
   - **c.** Pressing the Ctrl+Z key sequence once
   - **d.** Using the **quit** command

Foundation Topics

Accessing the Cisco Catalyst Switch CLI

Cisco uses the concept of a command-line interface (CLI) with its router and LAN switch products. The CLI is a text-based interface in which the user, typically a network engineer, enters a text command and presses Enter. Pressing Enter sends the command to the switch, which tells the device to do something. The switch does what the command says, and in some cases, the switch replies with some messages stating the results of the command.

Cisco Catalyst switches also support other methods to both monitor and configure a switch. For example, a switch can provide a web interface so that an engineer can open a web browser to connect to a web server running in the switch. Switches also can be controlled and operated using network management software.

This book discusses only Cisco Catalyst enterprise-class switches, and in particular, how to use the Cisco CLI to monitor and control these switches. This first major section of the chapter first examines these Catalyst switches in more detail and then explains how a network engineer can get access to the CLI to issue commands.

Cisco Catalyst Switches

When I was updating this chapter for expected publication in the year 2023, Cisco LAN switches fell into three product families:

- **Cisco Catalyst switches**, which serve as typical enterprise switches for use throughout an enterprise.
- **Cisco Nexus switches**, which Cisco designs for use in data centers. In comparison to Catalyst switches, Nexus switches support faster ports and more high-speed ports on a single switch, plus other features optimized for data centers.
- **Cisco Meraki switches**, built for enterprises, meet a need for LAN switches (and other network devices) that use a simple, cloud-based management portal that provides easy onboarding of new devices.

Throughout its history, when the CCNA exam mentions *switches*, it refers to Cisco Catalyst switches, and not to Nexus or Meraki switches.

Within the Cisco Catalyst brand of LAN switches, Cisco produces a wide variety of switch series or families. Each switch series includes several specific models of switches that have similar features, similar price-versus-performance tradeoffs, and similar internal components.

For example, when I was writing the latest version of this chapter, Cisco offered the Cisco Catalyst 9000 switch family as its primary Enterprise switch family. That family includes series that go by numbers like 9200, 9300, 9400, and so on, with specific models within each series. For instance, Cisco positions the 9200 and 9300 series of switches as full-featured access layer switches for enterprises.

Answers to the "Do I Know This Already?" quiz:

1 A, B **2** B **3** B **4** A **5** F **6** B, C

Figure 4-1 shows two models of 9200 switches. The lower switch in the figure has 48 fixed RJ-45 unshielded twisted-pair (UTP) 10/100/1000 ports, meaning that these ports can autonegotiate the use of 10BASE-T (10 Mbps), 100BASE-T (100 Mbps), or 1000BASE-T (1 Gbps) Ethernet. The upper switch has 24 such ports.

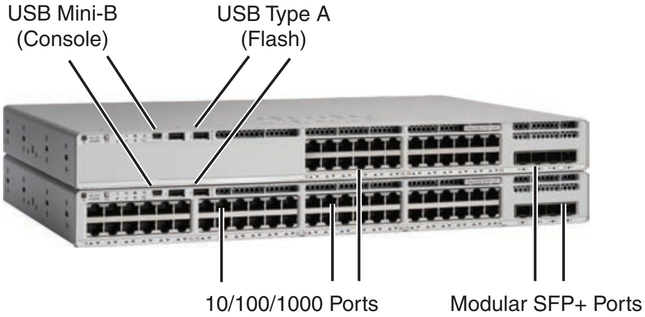


Figure 4-1 *Cisco Catalyst 9200L Switch Models, Front View*

Cisco refers to a switch's physical connectors as either *interfaces* or *ports*, with an interface type and interface number. Switches use interface types such as Ethernet, FastEthernet, GigabitEthernet, TenGigabit, and so on for faster speeds. For Ethernet interfaces that support running at multiple speeds, the switch uses the interface type of the fastest supported speed. For example, the switch refers to a 10/100/1000 interface (that is, an interface that runs at 10 Mbps, 100 Mbps, or 1000 Mbps) as a GigabitEthernet port, no matter the current speed used on the interface.

LAN switches also use a numeric interface identifier (interface ID) to identify the specific port. Interface IDs can use one, two, or three digits, with the digits separated by a slash, as chosen by Cisco when they designed the switch. For instance, Cisco Catalyst 9000 switches typically use three-digit identifiers like GigabitEthernet 1/0/1 and GigabitEthernet 1/0/2. Table 4-2 lists a few examples of interface identifiers and abbreviations for those identifiers based on the speeds supported by an interface.

Table 4-2 LAN Switch Interface ID Examples with Abbreviations

Speeds Supported	Common Name	Example Switch Interface ID	Valid Abbreviations
10 Mbps	Ethernet	Ethernet0/0	E0/0, Et0/0, Eth0/0
10/100 Mbps	10/100	FastEthernet 0/1	F0/1, Fa0/1
10/100/1000 Mbps	10/100/1000	GigabitEthernet 1/0/1	G1/0/1, Gi1/0/1
1G/2.5G/5G/10G	Multigig	TenGigabit 1/0/2	T1/0/2, Te1/0/2

Accessing the Cisco IOS XE CLI

The operating system (OS) in Catalyst switches creates a human-focused interface in which you can type commands and see lines of text in response. The next topic introduces that interface, called the command-line interface (CLI), but first the text gives a few background details about the operating system itself.

The Operating System in Cisco Catalyst Switches

Initially in its history, Cisco switches used an OS called CatOS, short for Catalyst OS. Cisco did not develop its first switch family, instead purchasing a company (Crescendo Communications) in 1993—a company that had named its switch family “Catalyst,” with the OS called CatOS.

When Cisco first got into the LAN switch business, Cisco routers used an OS written by Cisco, called Internetwork Operating System (**IOS**)—not IOS XE, simply IOS. Unsurprisingly, the router IOS CLI in Cisco routers worked differently than the switch CatOS CLI produced by the vendor that Cisco bought. Over time, Cisco created a switch OS based on IOS, so, mostly in the 1990s and 2000s, when Cisco released a new switch product family, it used IOS for Catalyst switches instead of CatOS. By moving its switches to use IOS, eventually all Cisco enterprise-class routers and switches used the same IOS CLI, making them easier to manage for networkers. Figure 4-2 depicts the migration timeline in general terms.

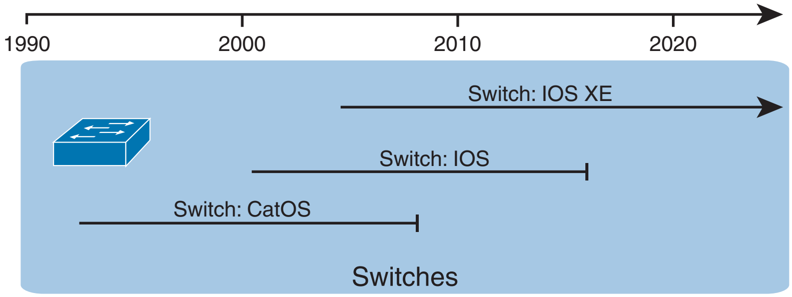


Figure 4-2 *Cisco Catalyst Operating System Long-Term Migration*

Cisco made yet another migration to a different OS for Catalyst switches, this time to use **IOS XE**. To create IOS XE, Cisco took IOS and modernized the internal software architecture. IOS XE, often referred to simply as XE, has features to improve uptime and the ability to maintain devices without requiring rebooting (reloading) the device. But it keeps the same familiar CLI; in fact, if you learned the CLI using a device running IOS, you might not even notice when later using a device running IOS XE.

For the purposes of CCNA, almost everything you see with commands and the CLI applies to older IOS-based switches as well as newer switches that use IOS XE. However, when a difference exists, the text will point out the difference.

> **NOTE** The book refers to IOS and IOS XE with the common term *IOS*, unless the need exists to differentiate between the two.

Accessing the IOS XE CLI

Cisco IOS Software for Catalyst switches implements and controls logic and functions performed by a Cisco switch. Besides controlling the switch’s performance and behavior, Cisco IOS also defines an interface for humans called the CLI. The Cisco IOS CLI allows the user to use a terminal emulation program, which accepts text entered by the user. When the user presses Enter, the terminal emulator sends that text to the switch. The switch processes the text as if it is a command, does what the command says, and sends text back to the terminal emulator.

The switch CLI can be accessed through three popular methods: the console, **Telnet**, and **Secure Shell (SSH)**. Two of these methods (Telnet and SSH) use the IP network in which the switch resides to reach the switch. The console is a physical port built specifically to allow access to the CLI. Figure 4-3 depicts the options.

Key Topic

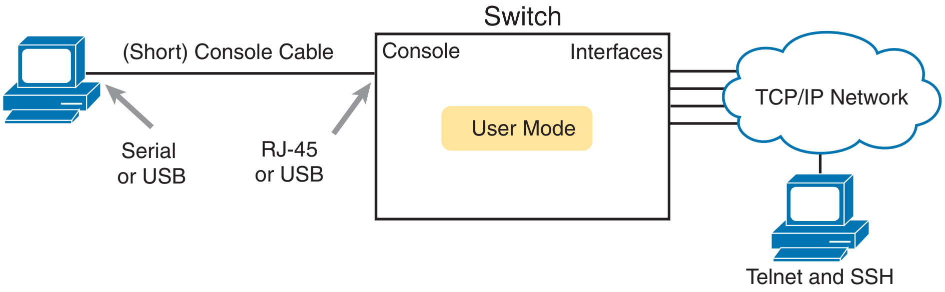


Figure 4-3 *CLI Access Options*

Console access requires both a physical connection between a PC (or other user device) and the switch's console port, as well as some software on the PC. Telnet and SSH require software on the user's device, but they rely on the existing TCP/IP network to transmit data. The next few pages detail how to connect the console and set up the software for each method to access the CLI.

Cabling the Console Connection

The physical console connection, both old and new, uses three main components: the physical console port on the switch, a physical serial port on the PC, and a cable that works with the console and serial ports. However, the physical cabling details have changed slowly over time, mainly because of advances and changes with serial interfaces on PC hardware. For this next topic, the text looks at three cases: newer connectors on both the PC and the switch, older connectors on both, and a third case with the newer (USB) connector on the PC but with an older connector on the switch.

Most PCs today use a familiar standard USB connector for the console connection. Cisco has been including USB console ports in new router and switch models for well over a decade. All you have to do is look at the switch to make sure you have the correct style of USB connector to match the USB console port (often a USB mini-B connector.) In the simplest form, you can use any USB port on the PC, with a USB cable, connected to the USB console port on the switch or router, as shown on the far-right side of Figure 4-4.

The case on the far left in the figure shows an older console connection, typical of how you would have connected to a switch over ten years ago. Before PCs used USB ports, they used serial ports for serial communications. The PC serial port had a D-shell connector (roughly rectangular) with nine pins (often called a DB-9). The console port looks like any Ethernet RJ-45 port (but is typically colored in blue and with the word *console* beside it on the switch). The older-style cabling used a standard RJ-45 to DB-9 converter plug and a UTP **rollover cable** with RJ-45 connectors on each end. The rollover pinout uses eight wires, rolling the wire at pin 1 to pin 8, pin 2 to pin 7, pin 3 to pin 6, and so on.

Key Topic

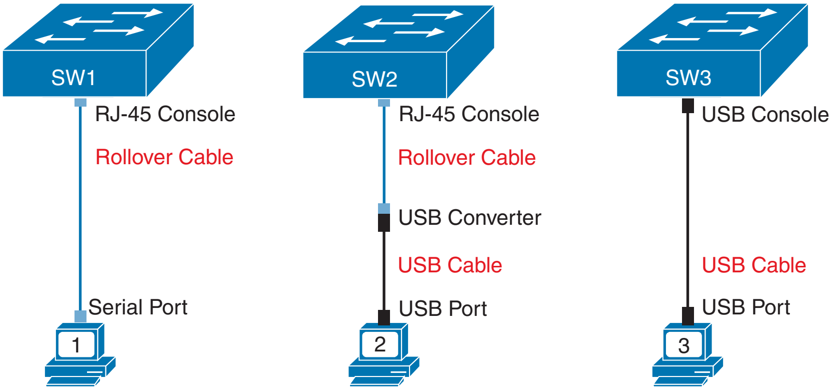


Figure 4-4 *Console Connection to a Switch*

The center case in the figure shows a variation that you might use on occasion that combines the cabling concepts from the left and right cases in the figure. You use the USB port on your PC but the RJ-45 console port on the switch. In fact, for some very old switch models, the switch has only an RJ-45 console port but no USB console port, requiring this kind of console cabling. In this case, you need a USB converter plug that converts from the older rollover console cable (with RJ-45 connectors) to a USB connector, as shown in the middle of Figure 4-4.

NOTE When using the USB options, you typically also need to install a software driver so that your PC's OS knows that the device on the other end of the USB connection is the console of a Cisco device. Also, you can easily find photos of these cables and components online, with searches like "cisco console cable," "cisco usb console cable," or "console cable converter."

Figure 4-5 shows a Cisco 9200L switch, rear view, which shows the RJ-45 console connector. The 9200L locates the USB console port (a mini-B USB port) on the front panel, as seen earlier in Figure 4-1.

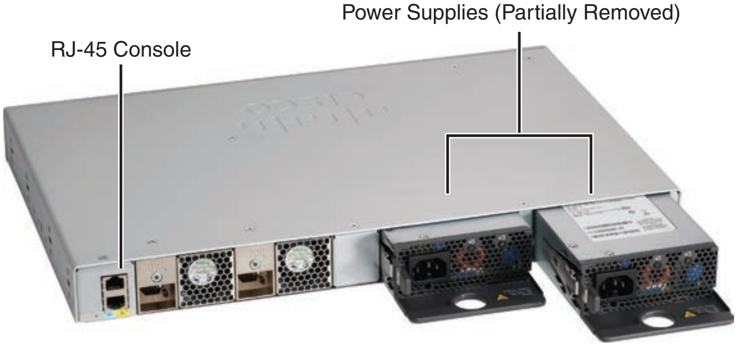


Figure 4-5 *Cisco Catalyst 9200L Switch, Rear View*

Configuring a Terminal Emulator

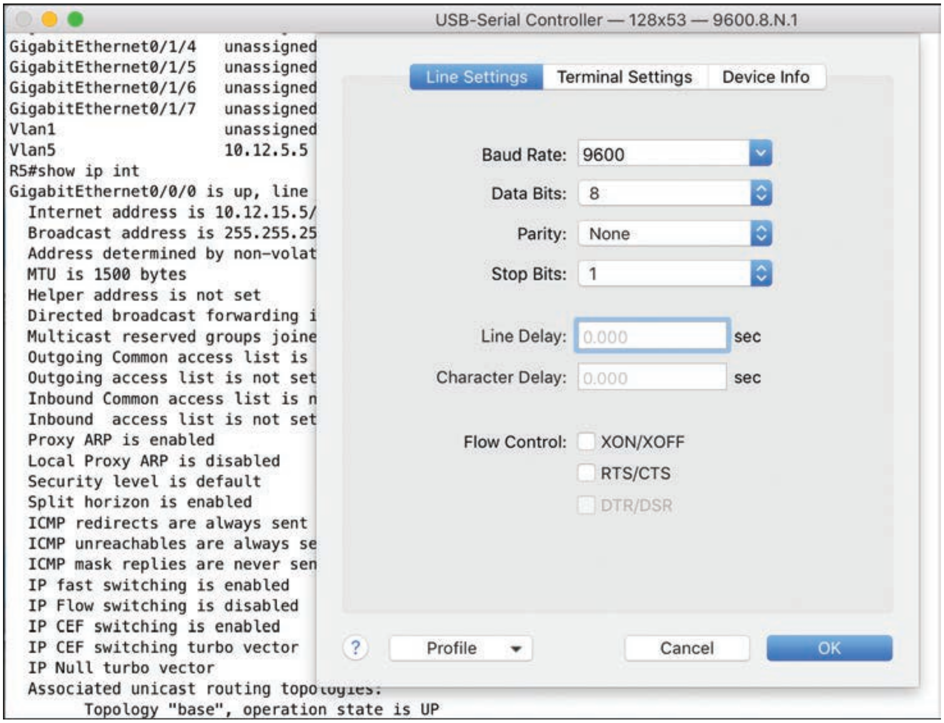
After the PC is physically connected to the console port, a terminal emulator software package must be installed and configured on the PC. The terminal emulator software treats all data as text. It accepts the text typed by the user and sends it over the console connection to the switch. Similarly, any bits coming into the PC over the console connection are displayed as text for the user to read.

The emulator must be configured to use the PC's serial port to match the settings on the switch's console port settings. The default console port settings on a switch are as follows. Note that the last three parameters are referred to collectively as 8N1:

Key Topic

- 9600 bits/second
- No hardware flow control
- 8-bit ASCII
- No parity bits
- 1 stop bit

Figure 4-6 shows one such terminal emulator. The image shows the window created by the emulator software in the background, with some output of a **show** command. The foreground, in the upper right, shows a settings window that lists the default console settings as listed just before this paragraph.


USB-Serial Controller — 128x53 — 9600.8.N.1

```
GigabitEthernet0/1/4   unassigned
GigabitEthernet0/1/5   unassigned
GigabitEthernet0/1/6   unassigned
GigabitEthernet0/1/7   unassigned
Vlan1                  unassigned
Vlan5                  10.12.5.5
R5#show ip int
GigabitEthernet0/0/0 is up, line
  Internet address is 10.12.15.5/
  Broadcast address is 255.255.25
  Address determined by non-volat
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding i
  Multicast reserved groups joine
  Outgoing Common access list is
  Outgoing access list is not set
  Inbound Common access list is n
  Inbound  access list is not set
  Proxy ARP is enabled
  Local Proxy ARP is disabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always se
  ICMP mask replies are never sen
  IP fast switching is enabled
  IP Flow switching is disabled
  IP CEF switching is enabled
  IP CEF switching turbo vector
  IP Null turbo vector
  Associated unicast routing topologies:
        Topology "base", operation state is UP
```

Baud Rate: 9600

Data Bits: 8

Parity: None

Stop Bits: 1

Line Delay: 0.000 sec

Character Delay: 0.000 sec

Flow Control: ☐ XON/XOFF ☐ RTS/CTS ☐ DTR/DSR

Figure 4-6 *Terminal Settings for Console Access*

Accessing the CLI with Telnet and SSH

For many years, terminal emulator applications have supported far more than the ability to communicate over a USB (or serial) port to a local device (like a switch's console). Terminal emulators support a variety of TCP/IP applications as well, including Telnet and SSH. Telnet and SSH both allow the user to connect to another device's CLI, but instead of connecting through a console cable to the console port, the traffic flows over the same IP network that the networking devices are helping to create.

Telnet uses the concept of a Telnet client (the terminal application) and a Telnet server (the switch in this case). A *Telnet client*, the device that sits in front of the user, accepts keyboard input and sends those commands to the *Telnet server*. The Telnet server accepts the text, interprets the text as a command, and replies back.

Cisco Catalyst switches enable a Telnet server by default, but switches need a few more configuration settings before you can successfully use Telnet to connect to a switch. Chapter 6, "Configuring Basic Switch Management," covers switch configuration to support Telnet and SSH in detail.

Using Telnet in a lab today makes sense, but Telnet poses a significant security risk in production networks. Telnet sends all data (including any username and password for login to the switch) as clear-text data. SSH gives us a much better option.

Think of SSH as the much more secure Telnet cousin. Outwardly, you still open a terminal emulator, connect to the switch's IP address, and see the switch CLI, no matter whether you use Telnet or SSH. The differences exist behind the scenes: SSH encrypts the contents of all messages, including the passwords, avoiding the possibility of someone capturing packets in the network and stealing the password to network devices.

User and Enable (Privileged) Modes

All three CLI access methods covered so far (console, Telnet, and SSH) place the user in an area of the CLI called *user EXEC mode*. User EXEC mode, sometimes also called **user mode**, allows the user to look around but not break anything. The "EXEC mode" part of the name refers to the fact that in this mode, when you enter a command, the switch executes the command and then displays messages that describe the command's results.

NOTE If you have not used the CLI before, you might want to experiment with the CLI from the Sim Lite product. You can find this resource on the companion website as mentioned in the Introduction.

Cisco IOS supports a more powerful EXEC mode called *privileged* mode (also known as **enable mode**). The formal name, *privileged mode*, refers to the fact that IOS allows powerful (or privileged) commands from that mode. Informally, engineers refer to the mode as *enable mode* because of the **enable** EXEC command, which moves the user from user mode to enable mode, as shown in Figure 4-7. For example, you can use the **reload** command, which tells the switch to reinitialize or reboot Cisco IOS, only from privileged mode.

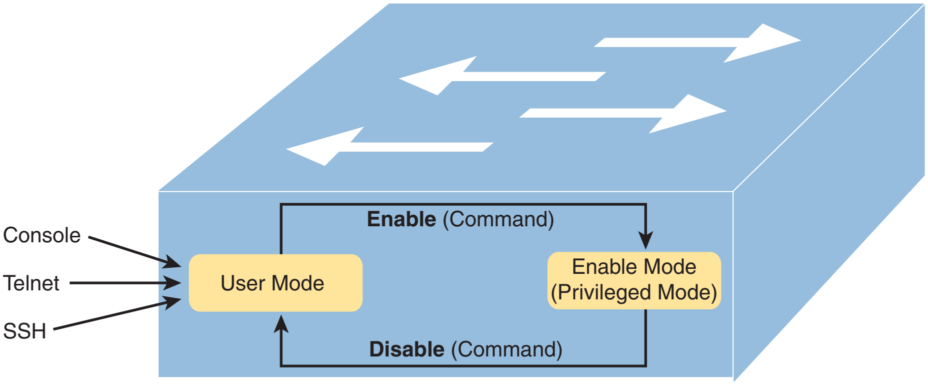


Figure 4-7 *User and Privileged Modes*

NOTE If the command prompt ends with a >, the user is in user mode; if it ends with a #, the user is in enable mode.

Example 4-1 demonstrates the differences between user and enable modes. The example shows the output that you could see in a terminal emulator window, for instance, when connecting from the console. In this case, the user sits at the user mode prompt ("Certskills1>") and tries the **reload** command. The **reload** command tells the switch to reinitialize or reboot Cisco IOS, so IOS allows this powerful command to be used only from enable mode. IOS rejects the **reload** command from user mode. Then the user moves to privileged (enable) mode using the **enable** EXEC command. At that point, IOS accepts the **reload** command now that the user is in enable mode.

Example 4-1 *Example of Privileged Mode Commands Being Rejected in User Mode*

```
Press RETURN to get started.


User Access Verification


Password:
Certskills1>
Certskills1> reload
Translating "reload"
% Unknown command or computer name, or unable to find computer address
Certskills1> enable
Password:
Certskills1#
Certskills1# reload


Proceed with reload? [confirm] y
00:08:42: %SYS-5-RELOAD: Reload requested by console. Reload Reason: Reload Command.
```

www.networkuser.ir

NOTE The commands that can be used in either user (EXEC) mode or enable (EXEC) mode are called EXEC commands.

This example is the first instance of this book showing you the output from the CLI, so it is worth noting a few conventions. The bold text represents what the user typed, and the nonbold text is what the switch sent back to the terminal emulator. Also, the typed passwords do not show up on the screen for security purposes.

Password Security for CLI Access from the Console

A Cisco switch, with default settings, remains relatively secure when locked inside a wiring closet, because by default, a switch allows console access only. By default, the console requires no password at all, and no password to reach enable mode for users that happened to connect from the console. The reason is that if you have access to the physical console port of the switch, you already have pretty much complete control over the switch. You could literally get out your screwdriver, remove the switch from the rack, and walk off with it, or you could unplug the power, or follow well-published procedures to go through password recovery to break into the CLI and then configure anything you want to configure.

However, many people use simple password protection for console users. Simple passwords can be configured at two points in the login process from the console: when the user connects from the console, and when any user moves to enable mode (using the **enable** EXEC command). You may have noticed that back in Example 4-1, the user saw a password prompt at both points.

Example 4-2 shows the additional configuration commands that were configured prior to collecting the output in Example 4-1. The output holds an excerpt from the EXEC command **show running-config**, which lists the current configuration in the switch.

Example 4-2 *Nondefault Basic Configuration*

```
Certskills1# show running-config
! Output has been formatted to show only the parts relevant to this discussion
hostname Certskills1
!
enable secret love
!
line console 0
 login

password faith
! The rest of the output has been omitted
Certskills1#
```

Working from top to bottom, note that the first configuration command listed by the **show running-config** command sets the switch's hostname to Certskills1. The switch uses the hostname to begin the command prompt.

Next, note that the lines with a **!** in them are comment lines, both in the text of this book and in the real switch CLI.

The **enable secret love** configuration command defines the password that all users must use to reach enable mode. So, no matter whether users connect from the console, Telnet, or SSH, they would use the password love when prompted for a password after typing the **enable** EXEC command.

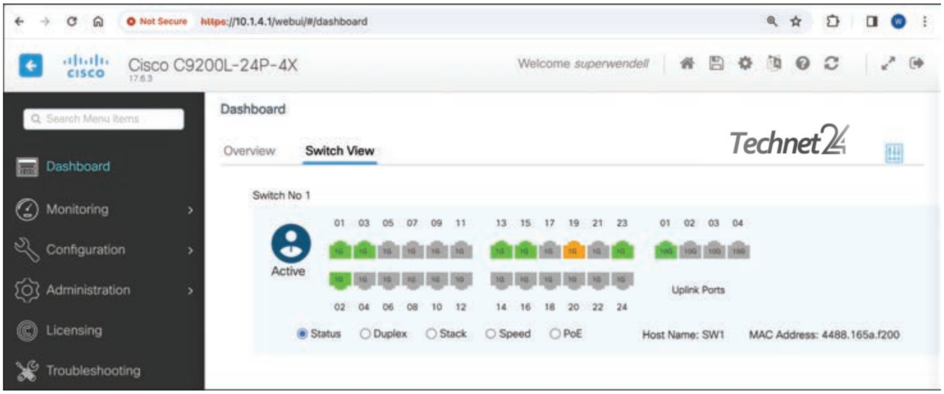
Finally, the last three lines configure the console password. The first line (**line console 0**) is the command that identifies the console, basically meaning "these next commands apply to the console only." The **login** command tells IOS to perform simple password checking (at the console). Remember, by default, the switch does not ask for a password for console users. Finally, the **password faith** command defines the password the console user must type when prompted.

This example just scratches the surface of the kinds of security configuration you might choose to configure on a switch, but it does give you enough detail to configure switches in your lab and get started (which is the reason I put these details in this first chapter of Part II). Note that Chapter 6 shows the configuration steps to add support for Telnet and SSH (including password security), and Chapter 9 of the *CCNA 200-301 Official Cert Guide, Volume 2*, Second Edition, "Securing Network Devices," shows additional security configuration as well.

Accessing the CLI with the WebUI

Engineers use SSH to access the network device CLI as a routine part of their daily work. Cisco also provides a graphical interface to manage individual Cisco switches and routers, referred to as the device's HTTP server or, in later years, as the WebUI (Web User Interface). Once configured, an engineer can use any web browser to connect to the IP address of the switch or router, supply the correct login information, and see web pages that allow management of that single device. That web interface also includes a method to access the CLI.

First, Figures 4-8 and 4-9 show a couple of example pages of the graphical interface. Figure 4-8 shows a screenshot of the web server dashboard's Switch View. This page shows an image representing the side of the switch, with all switch ports shown and colored lights representing the same status colors you would see on the physical switch.



Not Secure https://10.1.4.1/webui/#/dashboard

Cisco C9200L-24P-4X
17.6.3

Welcome *superwendell*

Host Name: SW1

MAC Address: 4488.165a.f200

Figure 4-8 *Cisco Switch Web Interface Dashboard: Switch View*

NOTE The examples in this section come from a Cisco Catalyst 9200L switch running IOS XE Version 17.6.3, visible in the upper left of the screenshot in Figure 4-8. However, the specifics of the user interface and available options in the left-side menus vary across device types, models, and IOS versions. So, use the figures in this section only to get a general idea of the functions of this tool.

The WebUI supports direct configuration and verification options that do not require knowledge of the CLI. For example, Figure 4-9 shows a screenshot with the user's mouse hovering over the word "Configuration" in the left menu, causing the page to reveal various configuration options. Earlier, the user had chosen the STP option, so the background in the screenshot shows a partial view of the Spanning Tree Protocol (STP) configuration screen. So, you could point and click through the user interface to configure some of the options covered for CCNA.

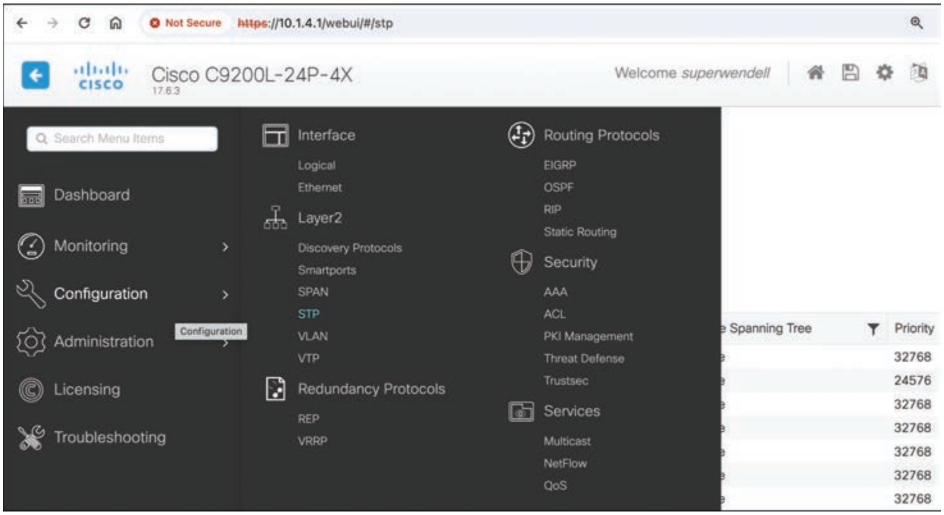
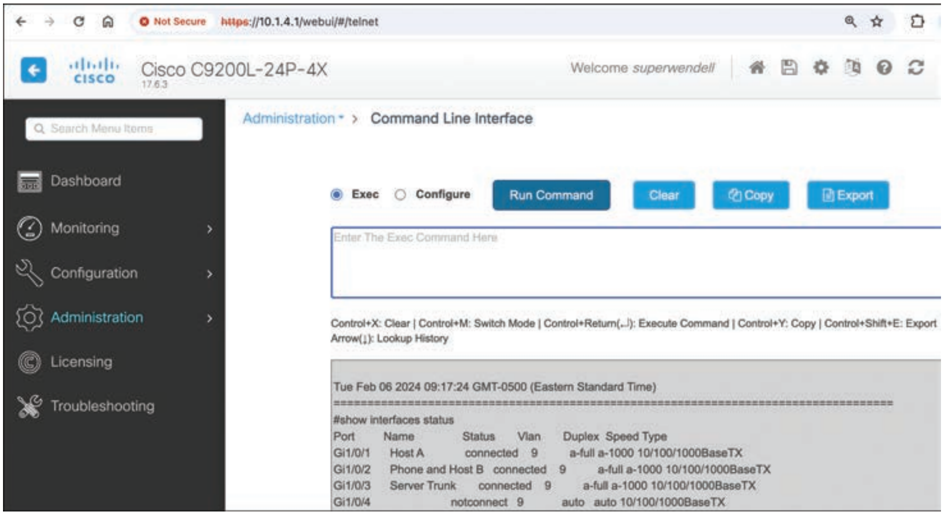


Figure 4-9 *Configuration Options in the Switch Web Server Configuration Menu*

If you have a Cisco switch or router available, you should take the time to use the WebUI and look around at some of the configuration and verification options. Again, be aware that your device might show different layouts and options versus the screenshots shown here.

The more recent versions of the WebUI provide access to the CLI without your device needing to have an SSH or Telnet client installed, as seen in Figure 4-10. Once you navigate to the correct page (**Administration > Command Line Interface**), you type the CLI command into the upper box, click a button to send it to the device, and the response messages appear in the lower box. The figure shows the output of the **show interfaces status** command.

The WebUI can be pretty useful when you do not have any of your devices with you. If your devices with SSH clients installed are back at your desk and you need to log in to a few network devices, borrow any device with a web browser, and connect to routers and switches. Although the WebUI is useful, most engineers prefer using an SSH client when working in the CLI because the clients are much more usable. For instance, in Figure 4-10, note the poor column alignment of the command output. An SSH client would typically not have those kinds of alignment issues, which can make reading the output more difficult.



Control+X: Clear | Control+M: Switch Mode | Control+Return(↵): Execute Command | Control+Y: Copy | Control+Shift+E: Export
Arrow(↓): Lookup History

```
Tue Feb 06 2024 09:17:24 GMT-0500 (Eastern Standard Time)
==================================================================================
#show interfaces status
Port      Name             Status      Vlan     Duplex  Speed Type
Gi1/0/1    Host A           connected    9        a-full a-1000 10/100/1000BaseTX
Gi1/0/2    Phone and Host B   connected    9        a-full a-1000 10/100/1000BaseTX
Gi1/0/3    Server Trunk      connected    9        a-full a-1000 10/100/1000BaseTX
Gi1/0/4                    notconnect   9        auto   auto 10/100/1000BaseTX
```

Figure 4-10 *CLI Access from the Switch Web Server Interface*

NOTE On a related note, vendors like Cisco also sell sophisticated stand-alone network management applications. These typically have graphical interfaces and comprehensive management for all available features and support all devices using one user interface. In comparison, the WebUI provides basic features on a per-device basis. You will learn more about one such application, Cisco Catalyst Center, in Chapter 22, "Cisco Software-Defined Access (Cisco SD-Access)," in the book *CCNA 200-301 Official Cert Guide, Volume 2*, Second Edition.

CLI Help Features

If you printed the Cisco IOS Command Reference documents, you would end up with a stack of paper several feet tall. No one should expect to memorize all the commands—and no one does. You can use several easy, convenient tools to help remember commands and save time typing. As you progress through your Cisco certifications, the exams will cover progressively more commands. However, you should know the methods of getting command help.

Table 4-3 summarizes command-recall help options available at the CLI. Note that, in the first column, *command* represents any command. Likewise, *parm* represents a command's parameter. For example, the third row lists *command* **?**, which means that commands such as **show ?** and **copy ?** would list help for the **show** and **copy** commands, respectively.

Table 4-3 Cisco IOS Software Command Help

What You Enter	What Help You Get
?	This option provides help for all commands available in this mode.
command ?	With a space between the command and the ?, the switch lists text to describe all the first parameter options for the command.
com?	This option lists commands that start with **com**.
command parm?	This option lists all parameters beginning with the **parameter typed so far.** (Notice that there is no space between *parm* and the **?**.)

What You Enter	What Help You Get
command parm<Tab>	Pressing the Tab key causes IOS to spell out the rest of the word, assuming that you have typed enough of the word so there is only one option that begins with that string of characters.
command parm1 **?**	If a space is inserted before the question mark, the CLI lists all the next parameters and gives a brief explanation of each.

When you enter the **?**, the Cisco IOS CLI reacts immediately; that is, you don't need to press the Enter key or any other keys. The device running Cisco IOS also redisplays what you entered before the **?** to save you some keystrokes. If you press Enter immediately after the **?**, Cisco IOS tries to execute the command with only the parameters you have entered so far.

The information supplied by using help depends on the CLI mode. For example, when **?** is entered in user mode, the commands allowed in user mode are displayed, but commands available only in enable mode (not in user mode) are not displayed. Also, help is available in **configuration mode**, which is the mode used to configure the switch. In fact, configuration mode has many different subconfiguration modes, as explained in the section "Configuration Submodes and Contexts," later in this chapter. So, you can get help for the commands available in each configuration submode as well. (Note that this might be a good time to use the free Sim Lite product on the companion website: to do so, open any lab, use the question mark, and try some commands.)

Cisco IOS stores the commands that you enter in a history buffer, storing ten commands by default. The CLI allows you to move backward and forward in the historical list of commands and then edit the command before reissuing it. These key sequences can help you use the CLI more quickly on the exams. Table 4-4 lists the commands used to manipulate previously entered commands.

Table 4-4 Key Sequences for Command Edit and Recall

Keyboard Command	What Happens
Up arrow or Ctrl+P	This keyboard command displays the most recently used command. If you press it again, the next most recent command appears, until the history buffer is exhausted. (The *P* stands for previous.)
Down arrow or Ctrl+N	If you have gone too far back into the history buffer, these keys take you forward to the more recently entered commands. (The *N* stands for next.)
Left arrow or Ctrl+B	This keyboard command moves the cursor backward in the currently displayed command without deleting characters. (The *B* stands for back.)
Right arrow or Ctrl+F	This keyboard command moves the cursor forward in the currently displayed command without deleting characters. (The *F* stands for forward.)
Backspace	This keyboard command moves the cursor backward in the currently displayed command, deleting characters.

The debug and show Commands

By far, the single most popular Cisco IOS command is the **show** command. The **show** command has a large variety of options, and with those options, you can find the status of almost every feature of Cisco IOS. Essentially, the **show** command lists the currently known facts about the switch's operational status.

For example, consider the output from the **show mac address-table dynamic** command listed in Example 4-3. This **show** command, issued from user mode, lists the table the switch uses to make forwarding decisions. A switch's MAC address table basically lists the data that a switch uses to do its primary job.

Example 4-3 *Example MAC Address Table*

```
Certskills1> show mac address-table dynamic
 Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  31    0200.1111.1111    DYNAMIC     Gi0/1
  31    0200.3333.3333    DYNAMIC     Fa0/3
  31    1833.9d7b.0e9a    DYNAMIC     Gi0/1
  10    1833.9d7b.0e9a    DYNAMIC     Gi0/1
  10    30f7.0d29.8561    DYNAMIC     Gi0/1
   1    1833.9d7b.0e9a    DYNAMIC     Gi0/1
  12    1833.9d7b.0e9a    DYNAMIC     Gi0/1
Total Mac Addresses for this criterion: 7
Certskills1>
```

The **debug** command also tells the user details about the operation of the switch. However, while the **show** command lists status information at one instant of time—more like a photograph—the **debug** command acts more like a live video camera feed. Once you issue a **debug** command, IOS remembers, issuing messages over time as events continue to occur. Any switch user can choose to receive those messages, with the switch sending the messages to the console by default. Most of the commands used throughout this book to verify operation of switches and routers are **show** commands.

Configuring Cisco IOS Software

You will want to configure every switch in an Enterprise network, even though the switches will forward traffic even with default configuration. This section covers the basic configuration processes, including the concept of a configuration file and the locations in which the configuration files can be stored. Although this section focuses on the configuration process, and not on the configuration commands themselves, you should know all the commands covered in this chapter for the exams, in addition to the configuration processes.

Configuration mode accepts *configuration commands*—commands that tell the switch the details of what to do and how to do it. User and privileged modes accept EXEC commands, which return output, or possibly take an action like reloading the switch, but commands in these modes do not change any configuration settings. Figure 4-11 illustrates the navigation among configuration mode, user EXEC mode, and privileged EXEC mode.

Commands entered in configuration mode update the active configuration file. *These changes to the configuration occur immediately each time you press the Enter key at the end of a command.* Be careful when you enter a configuration command!

Key Topic

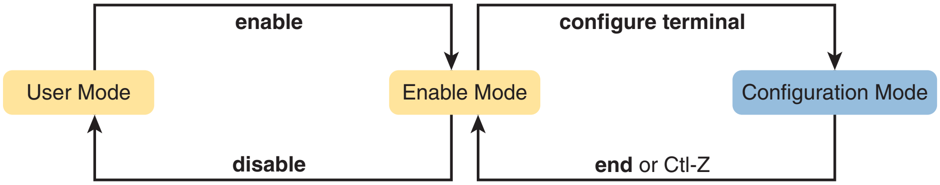


Figure 4-11 *CLI Configuration Mode Versus EXEC Modes*

Configuration Submodes and Contexts

Configuration mode supports a multitude of commands. To help organize the configuration, IOS groups some kinds of configuration commands together. To do that, when using configuration mode, you move from the initial mode—global configuration mode—into subcommand modes. *Context-setting commands* move you from one configuration subcommand mode, or context, to another. These context-setting commands tell the switch the topic about which you will enter the next few configuration commands. More importantly, the context tells the switch the topic you care about right now, so when you use the **?** to get help, the switch gives you help about that topic only.

NOTE *Context-setting* is not a Cisco term. It is just a description used here to help make sense of configuration mode.

The best way to learn about configuration submodes is to use them, but first, take a look at these upcoming examples. For instance, the **interface** command is one of the most commonly used context-setting configuration commands. For example, the CLI user could enter interface configuration mode by entering the **interface FastEthernet 0/1** configuration command. Asking for help in interface configuration mode displays only commands that are useful when configuring Ethernet interfaces. Commands used in this context are called *subcommands*—or, in this specific case, *interface subcommands*. When you begin practicing with the CLI using real equipment, the navigation between modes can become natural. For now, consider Example 4-4, which shows the following:

- Movement from enable mode to global configuration mode by using the **configure terminal** EXEC command.
- Use of a **hostname Fred** global configuration command to configure the switch's name. Using a global command from global configuration mode leaves you in global configuration mode.
- Movement from global configuration mode to console line configuration mode (using the **line console 0** command). The **line** command is another of the small set of context-setting commands that move you to another submode.

- Setting the console's simple password to **hope** (using the **password hope** line subcommand). Using a subcommand while in that submode leaves the command prompt in that submode.
- Movement from console configuration mode to interface configuration mode (using the **interface** *type number* command). The **interface** command is another of the small set of context-setting commands that move you to another submode.
- Setting the speed to 100 Mbps for interface Fa0/1 (using the **speed 100** interface subcommand).
- Movement from interface configuration mode back to global configuration mode (using the **exit** command).

Example 4-4 *Navigating Between Different Configuration Modes*

```
Switch# configure terminal
Switch(config)# hostname Fred
Fred(config)# line console 0
Fred(config-line)# password hope
Fred(config-line)# interface FastEthernet 0/1
Fred(config-if)# speed 100
Fred(config-if)# exit
Fred(config)#
```

The text inside parentheses in the command prompt identifies the configuration mode. For example, the first command prompt after you enter configuration mode lists (config), meaning global configuration mode. After the **line console 0** command, the text expands to (config-line), meaning line configuration mode. Each time the command prompt changes within config mode, you have moved to another configuration mode.

Table 4-5 shows the most common command prompts in configuration mode, the names of those modes, and the context-setting commands used to reach those modes.

Key Topic

Table 4-5 Common Switch Configuration Modes

Prompt	Name of Mode	Context-Setting Command(s) to Reach This Mode
hostname(config)#	Global	None—first mode after **configure terminal**
hostname(config-line)#	Line	**line console 0** **line vty 0 15**
hostname(config-if)#	Interface	**interface** *type number*
hostname(config-vlan)#	VLAN	**vlan** *number*

You should practice until you become comfortable moving between the different configuration modes, back to enable mode, and then back into the configuration modes. However, you can learn these skills just doing labs about the topics in later chapters of the book. For now, Figure 4-12 shows most of the navigation between global configuration mode and the four configuration submodes listed in Table 4-5.

Key Topic

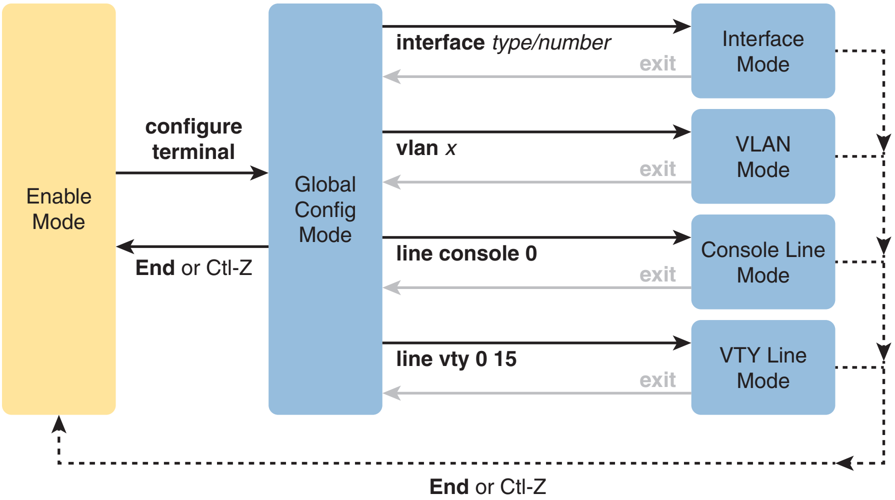


Figure 4-12 *Navigation In and Out of Switch Configuration Modes*

You really should stop and try navigating around these configuration modes. If you have not yet decided on a lab strategy, install the Pearson Sim Lite software from the companion website. It includes the simulator and a couple of lab exercises. Start any lab, ignore the instructions, and just get into configuration mode and move around between the configuration modes shown in Figure 4-12.

No set rules exist for what commands happen to be global commands or subcommands. Instead, you learn new commands, see whether they are global commands, or if subcommands, you also learn the required mode. But generally, Cisco uses global commands for settings that apply to the entire switch and subcommands that apply to one component or feature. For example:

The global command **hostname** sets the one hostname for the entire switch.

The interface subcommand **speed** configures a setting for a specific interface, so it works only in interface configuration submode.

Storing Switch Configuration Files

When you configure a switch, it needs to use the configuration. It also needs to be able to retain the configuration in case the switch loses power. Cisco switches contain random-access memory (RAM) to store data while Cisco IOS is using it, but RAM loses its contents when the switch loses power or is reloaded. To store information that must be retained when the switch loses power or is reloaded, Cisco switches use several types of more permanent memory, none of which has any moving parts. By avoiding components with moving parts (such as traditional disk drives), switches can maintain better uptime and availability.

The following list details the four main types of memory found in Cisco switches, as well as the most common use of each type:

- **RAM:** Sometimes called DRAM, for dynamic random-access memory, RAM is used by the switch just as it is used by any other computer: for working storage. The running (active) configuration file is stored here.

- **Flash memory:** Either a chip inside the switch or a removable memory card, flash memory stores fully functional Cisco IOS images and is the default location where the switch gets its Cisco IOS at boot time. Flash memory also can be used to store any other files, including backup copies of configuration files.
- **ROM:** Read-only memory (ROM) stores a bootstrap (or boothelper) program that is loaded when the switch first powers on. This bootstrap program then finds the full Cisco IOS image and manages the process of loading Cisco IOS into RAM, at which point Cisco IOS takes over operation of the switch.
- **NVRAM:** Nonvolatile RAM (NVRAM) stores the initial or startup configuration file that is used when the switch is first powered on and when the switch is reloaded.

Figure 4-13 summarizes this same information in a briefer and more convenient form for memorization and study.

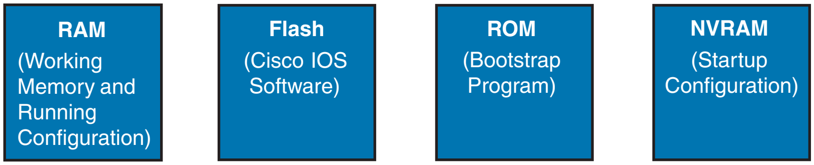


Figure 4-13 *Cisco Switch Memory Types*

Cisco IOS stores the collection of configuration commands in a *configuration file*. In fact, switches use multiple configuration files—one file for the initial configuration used when powering on, and another configuration file for the active, currently used running configuration as stored in RAM. Table 4-6 lists the names of these two files, their purpose, and their storage location.

Key Topic

Table 4-6 Names and Purposes of the Two Main Cisco IOS Configuration Files

Configuration Filename	Purpose	Where It Is Stored
startup-config	Stores the initial configuration used anytime the switch reloads Cisco IOS.	NVRAM
running-config	Stores the currently used configuration commands. This file changes dynamically when someone enters commands in configuration mode.	RAM

Essentially, when you use configuration mode, you change only the **running-config file**. This means that the configuration example earlier in this chapter (Example 4-4) updates only the running-config file. However, if the switch lost power right after that example, all that configuration would be lost. If you want to keep that configuration, you have to copy the running-config file into NVRAM, overwriting the old **startup-config file**.

Example 4-5 demonstrates that commands used in configuration mode change only the running configuration in RAM. The example shows the following concepts and steps:

Step 1. The example begins with both the running and startup-config having the same hostname, per the **hostname hannah** command.

Step 2. The hostname is changed in configuration mode using the **hostname harold** command.

Step 3. The **show running-config** and **show startup-config** commands show the fact that the hostnames are now different, with the **hostname harold** command found only in the running-config.

Example 4-5 *How Configuration Mode Commands Change the Running-Config File, Not the Startup-Config File*

```
! Step 1 next (two commands)
!
hannah# show running-config
! (lines omitted)
hostname hannah
! (rest of lines omitted)

hannah# show startup-config
! (lines omitted)
hostname hannah
! (rest of lines omitted)
! Step 2 next. Notice that the command prompt changes immediately after
! the hostname command.

hannah# configure terminal
hannah(config)# hostname harold
harold(config)# exit
! Step 3 next (two commands)
!
harold# show running-config
! (lines omitted) - just showing the part with the hostname command
hostname harold
!
harold# show startup-config
! (lines omitted) - just showing the part with the hostname command
hostname hannah
```

Copying and Erasing Configuration Files

The configuration process updates the running-config file, which is lost if the router loses power or is reloaded. Clearly, IOS needs to provide us a way to copy the running configuration so that it will not be lost, so it will be used the next time the switch reloads or powers on. For instance, Example 4-5 ended with a different running configuration (with the **hostname harold** command) versus the startup configuration.

In short, the EXEC command **copy running-config startup-config** backs up the running-config to the startup-config file. This command overwrites the current startup-config file with what is currently in the running-config file.

In addition, in the lab, you may want to just get rid of all existing configuration and start over with a clean configuration. To do that, you can erase the startup-config file using three different commands:

```
write erase
erase startup-config
erase nvram:
```

4

Once the startup-config file is erased, you can reload or power off/on the switch, and it will boot with the now-empty startup configuration.

Note that Cisco IOS does not have a command that erases the contents of the running-config file. To clear out the running-config file, simply erase the startup-config file, and then **reload** the switch, and the running-config will be empty at the end of the process.

NOTE Cisco uses the term *reload* to refer to what most PC operating systems call rebooting or restarting. In each case, it is a re-initialization of the software. The **reload** EXEC command causes a switch to reload.

Chapter Review

One key to doing well on the exams is to perform repetitive spaced review sessions. Review this chapter's material using either the tools in the book or on the book's companion website. Refer to the "Your Study Plan" element section titled "Step 2: Build Your Study Habits Around the Chapter" for more details. Table 4-7 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 4-7 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP
Review memory tables		Book, website
Review command tables		Book

Review All the Key Topics

Key Topic

Table 4-8 Key Topics for Chapter 4

Key Topic Element	Description	Page Number
Figure 4-3	Three methods to access a switch CLI	93
Figure 4-4	Cabling options for a console connection	94
List	A Cisco switch's default console port settings	95
Figure 4-11	Navigation between user, enable, and global config modes	104
Table 4-5	A list of configuration mode prompts, the name of the configuration mode, and the command used to reach each mode	105
Figure 4-12	Configuration mode context-setting commands	106
Table 4-6	The names and purposes of the two configuration files in a switch or router	107

Key Terms You Should Know

command-line interface (CLI), configuration mode, enable mode, IOS, IOS XE, rollover cable, running-config file, Secure Shell (SSH), startup-config file, Telnet, user mode

Command References

Tables 4-9 and 4-10 list configuration and verification commands used in this chapter, respectively. As an easy review exercise, cover the left column in a table, read the right column, and try to recall the command without looking. Then repeat the exercise, covering the right column, and try to recall what the command does.

Table 4-9 Chapter 4 Configuration Commands

Command	Mode and Purpose
line console 0	Global command that changes the context to console configuration mode.
login	Line (console and vty) configuration mode. Tells IOS to prompt for a password (no username).
password *pass-value*	Line (console and vty) configuration mode. Sets the password required on that line for login if the **login** command (with no other parameters) is also configured.
interface *type port-number*	Global command that changes the context to interface mode—for example, **interface FastEthernet 0/1**.
speed *value*	Interface subcommand that sets the Ethernet interface speed on interfaces that support multiple speeds.
hostname *name*	Global command that sets this switch's hostname, which is also used as the first part of the switch's command prompt.
exit	Command that moves back to the next higher mode in configuration mode.

Command	Mode and Purpose
end	Command that exits configuration mode and goes back to enable mode from any of the configuration submodes.
Ctrl+Z	This is not a command, but rather a two-key combination (pressing the Ctrl key and the letter Z) that together do the same thing as the **end** command.

Table 4-10 Chapter 4 EXEC Command Reference

Command	Purpose
no debug all **undebug all**	Privileged mode EXEC command to disable all currently enabled debugs.
reload	Privileged mode EXEC command that reboots the switch or router.
copy running-config startup-config	Privileged mode EXEC command that saves the active config, replacing the startup-config file used when the switch initializes.
copy startup-config running-config	Privileged mode EXEC command that merges the startup-config file with the currently active config file in RAM.
show running-config	Privileged mode EXEC command that lists the contents of the running-config file.
write erase **erase startup-config** **erase nvram:**	Privileged mode EXEC command that erases the startup-config file.
quit	EXEC command that disconnects the user from the CLI session.
show startup-config	Privileged mode EXEC command that lists the contents of the startup-config (initial config) file.
enable	User mode EXEC command that moves the user from user mode to enable (privileged) mode and prompts for a password if one is configured.
disable	Privileged mode EXEC command that moves the user from privileged mode to user mode.
configure terminal	Privileged mode EXEC command that moves the user into configuration mode.
show mac address-table	EXEC command that lists the contents of a switch forwarding (MAC) table.

CHAPTER 5

Analyzing Ethernet LAN Switching

This chapter covers the following exam topics:

- **1.0 Network Fundamentals**
  - **1.1 Explain the role and function of network components**
    - **1.1.b Layer 2 and Layer 3 switches**
  - **1.13 Describe switching concepts**
    - **1.13.a MAC learning and aging**
    - **1.13.b Frame switching**
    - **1.13.c Frame flooding**
    - **1.13.d MAC address table**
- **2.0 Network Access**
  - **2.5 Interpret basic operations of Spanning Tree Protocols**

When you buy a Cisco Catalyst Ethernet switch, the switch is ready to work. All you have to do is take it out of the box, power on the switch by connecting the power cable to the switch and a power outlet, and connect hosts to the switch using the correct unshielded twisted-pair (UTP) cables. You do not have to configure anything else, or connect to the console and login, or do anything: the switch just starts forwarding Ethernet frames.

In Part II of this book, you will learn how to build, configure, and verify the operation of Ethernet LANs. In Chapter 4, “Using the Command-Line Interface,” you learned how to move around in the CLI, issue commands, and configure the switch. This chapter takes a short but important step in that journey by explaining the logic a switch uses when forwarding Ethernet frames.

This chapter breaks the content into two major sections. The first reviews and then further develops the concepts behind LAN switching, which were first introduced back in Chapter 2, “Fundamentals of Ethernet LANs.” The second section then uses IOS **show** commands to verify that Cisco switches actually learned the MAC addresses, built its MAC address table, and forwarded frames.

“Do I Know This Already?” Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book as well as on the companion website, includes both the answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 5-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
LAN Switching Concepts	1–4
Verifying and Analyzing Ethernet Switching	5–6

1. Which of the following statements describes part of the process of how a switch decides to forward a frame destined for a known unicast MAC address?
   - **a.** It compares the unicast destination address to the bridging, or MAC address, table.
   - **b.** It compares the unicast source address to the bridging, or MAC address, table.
   - **c.** It forwards the frame out all interfaces in the same VLAN except for the incoming interface.
   - **d.** It compares the destination IP address to the destination MAC address.
   - **e.** It compares the frame's incoming interface to the source MAC entry in the MAC address table.
2. Which of the following statements best describes the forwarding logic that a LAN switch, with all interfaces assigned to VLAN 1 as per default settings, uses for an incoming frame with a destination MAC address of FFFF.FFFF.FFFF?
   - **a.** It forwards the frame out all switch ports.
   - **b.** It forwards the frame out all switch ports except the arrival port.
   - **c.** It forwards the frame out all ports that had earlier registered to ask to receive broadcasts.
   - **d.** It discards the frame.
3. Which of the following statements best describes what a switch does with a frame destined for an unknown unicast address?
   - **a.** It forwards out all interfaces in the same VLAN except for the incoming interface.
   - **b.** It forwards the frame out the one interface identified by the matching entry in the MAC address table.
   - **c.** It compares the destination IP address to the destination MAC address.
   - **d.** It compares the frame's incoming interface to the source MAC entry in the MAC address table.
4. Which of the following comparisons does a switch make when deciding whether a new MAC address should be added to its MAC address table?
   - **a.** It compares the unicast destination address to the bridging, or MAC address, table.
   - **b.** It compares the unicast source address to the bridging, or MAC address, table.
   - **c.** It compares the VLAN ID to the bridging, or MAC address, table.
   - **d.** It compares the destination IP address's ARP cache entry to the bridging, or MAC address, table.

5. A Cisco Catalyst switch has 24 10/100 ports, numbered 0/1 through 0/24. Ten PCs connect to the ten lowest numbered ports, with those PCs working and sending data over the network. The other ports are not connected to any device. Which of the following answers lists facts displayed by the **show interfaces status** command?
   - **a.** Port Ethernet 0/1 is in a connected state.
   - **b.** Port Fast Ethernet 0/11 is in a connected state.
   - **c.** Port Fast Ethernet 0/5 is in a connected state.
   - **d.** Port Ethernet 0/15 is in a notconnected state.

6. Consider the following output from a Cisco Catalyst switch:

```
SW1# show mac address-table dynamic
          Mac Address Table
------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1     02AA.AAAA.AAAA    DYNAMIC     Gi0/1
   1     02BB.BBBB.BBBB    DYNAMIC     Gi0/2
   1     02CC.CCCC.CCCC    DYNAMIC     Gi0/3
Total Mac Addresses for this criterion: 3
```

   Which of the following answers is true about this switch?
   - **a.** The output proves that port Gi0/2 connects directly to a device that uses address 02BB.BBBB.BBBB.
   - **b.** The switch has learned three MAC addresses since the switch powered on.
   - **c.** The three listed MAC addresses were learned based on the destination MAC address of frames forwarded by the switch.
   - **d.** Address 02CC.CCCC.CCCC was learned from the source MAC address of a frame that entered port Gi0/3.

Foundation Topics

LAN Switching Concepts

A modern Ethernet LAN connects user devices as well as servers into some switches, with the switches then connecting to each other, sometimes in a design like Figure 5-1. Part of the LAN, called a campus LAN, supports the end-user population as shown on the left of the figure. End-user devices connect to LAN switches, which in turn connect to other switches so that a path exists to the rest of the network. The campus LAN switches sit in wiring closets close to the end users. On the right, the servers used to provide information to the users also connect to the LAN. Those servers and switches often sit in a closed room called a *data center*, with connections to the campus LAN to support traffic to/from the users.

To forward traffic from a user device to a server and back, each switch performs the same kind of logic, independently from each other. The first half of this chapter examines the logic—how a switch chooses to forward an Ethernet frame, when the switch chooses to not forward the frame, and so on.

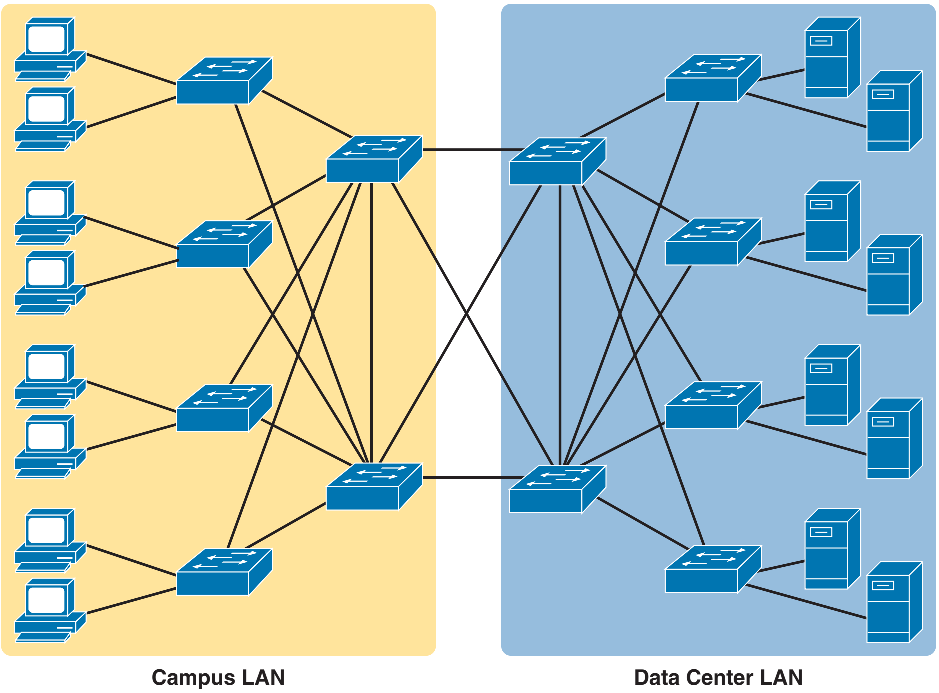


Figure 5-1 *Campus LAN and Data Center LAN, Conceptual Drawing*

Overview of Switching Logic

Ultimately, the role of a LAN switch is to forward Ethernet frames. LANs exist as a set of user devices, servers, and other devices that connect to switches, with the switches connected to each other. The LAN switch has one primary job: to forward frames to the correct destination (MAC) address. And to achieve that goal, switches use logic—logic based on the source and destination MAC address in each frame's Ethernet header.

LAN switches receive Ethernet frames and then make a switching decision: either forward the frame out some other ports or ignore the frame. To accomplish this primary mission, switches perform three actions:

Key Topic

1. Deciding when to forward a frame or when to filter (not forward) a frame, based on the destination MAC address
2. Preparing to forward future frames by learning the source MAC address of each frame received by the switch
3. Cooperating with all switches to prevent the endless looping of frames by using Spanning Tree Protocol (STP)

The first action is the switch's primary job, whereas the other two items are overhead functions.

NOTE Throughout this book's discussion of LAN switches, the terms *switch port* and *switch interface* are synonymous.

Most of the upcoming discussions and figures about Ethernet switching focus on the use of the ever-present destination and source MAC address fields in the header. Both are 6 bytes long (represented as 12 hex digits in the book) and are a key part of the switching logic discussed in this section. Refer back to Chapter 2's section titled "Ethernet Data-Link Protocols" for a discussion of the header in detail for more info on the rest of the Ethernet frame. Figure 5-2 repeats the frame format here for reference.

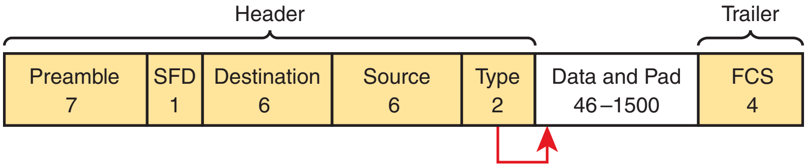


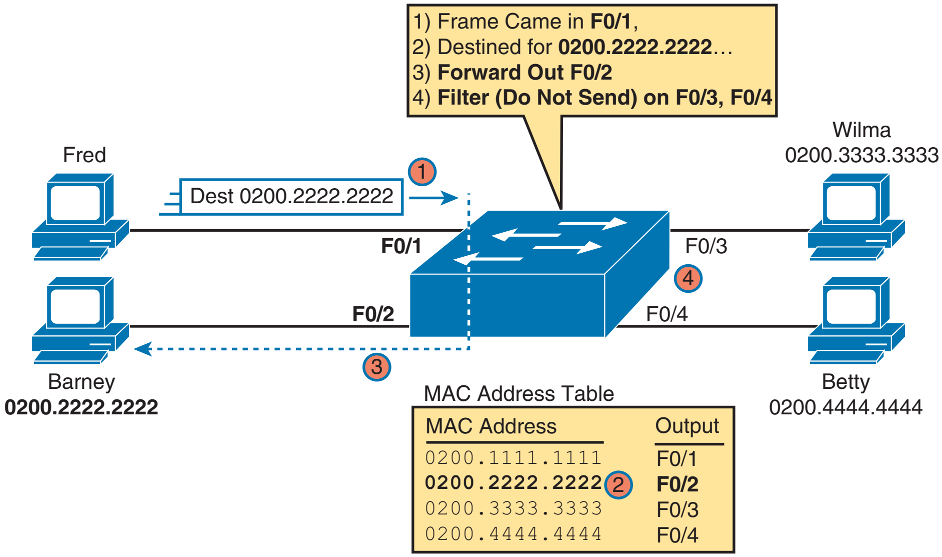
Figure 5-2 *IEEE 802.3 Ethernet Frame (One Variation)*

Now on to the details of how Ethernet switching works!

Forwarding Known Unicast Frames

To decide whether to forward a frame, a switch uses a dynamically built table that lists MAC addresses and outgoing interfaces. Switches compare the frame's destination MAC address to this table to decide whether the switch should forward a frame or simply ignore it. For example, consider the simple network shown in Figure 5-3, with Fred sending a frame to Barney.

Key Topic



1) Frame Came in **F0/1**,
2) Destined for **0200.2222.2222**…
3) **Forward Out F0/2**
4) **Filter (Do Not Send) on F0/3, F0/4**

MAC Address Table

MAC Address	Output
0200.1111.1111	F0/1
0200.2222.2222	**F0/2**
0200.3333.3333	F0/3
0200.4444.4444	F0/4

Figure 5-3 *Sample Switch Forwarding and Filtering Decision*

Answers to the "Do I Know This Already?" quiz:

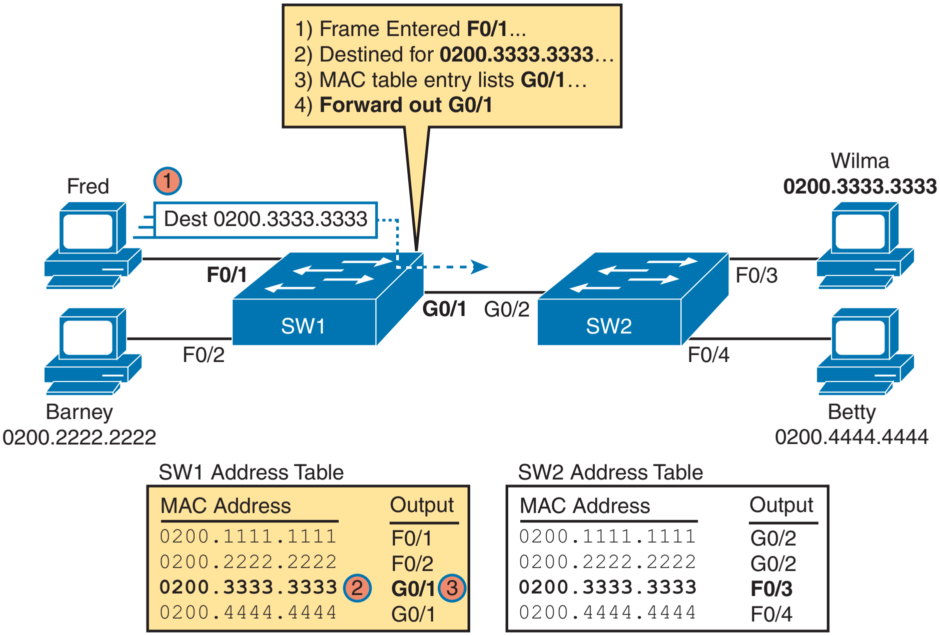
1 A **2** B **3** A **4** B **5** C **6** D

In this figure, Fred sends a frame with destination address 0200.2222.2222 (Barney's MAC address). The switch compares the destination MAC address (0200.2222.2222) to the MAC address table, matching the bold table entry. That matched table entry tells the switch to forward the frame out port F0/2, and only port F0/2.

NOTE A switch's **MAC address table** is also called the *switching table*, or *bridging table*, or even the *content-addressable memory (CAM) table*, in reference to the type of physical memory used to store the table.

A switch's MAC address table lists the location of each MAC relative to that one switch. In LANs with multiple switches, each switch makes an independent forwarding decision based on its own MAC address table. Together, they forward the frame so that it eventually arrives at the destination.

For example, Figure 5-4 shows the first switching decision in a case in which Fred sends a frame to Wilma, with destination MAC 0200.3333.3333. The topology has changed versus the previous figure, this time with two switches, and Fred and Wilma connected to two different switches. Figure 5-4 shows the first switch's logic, in reaction to Fred sending the original frame. Basically, the switch receives the frame in port F0/1, finds the destination MAC (0200.3333.3333) in the MAC address table, sees the outgoing port of G0/1, so SW1 forwards the frame out its G0/1 port.



1) Frame Entered **F0/1**...
2) Destined for **0200.3333.3333**...
3) MAC table entry lists **G0/1**...
4) **Forward out G0/1**

SW1 Address Table

MAC Address	Output
0200.1111.1111	F0/1
0200.2222.2222	F0/2
0200.3333.3333 ②	**G0/1** ③
0200.4444.4444	G0/1

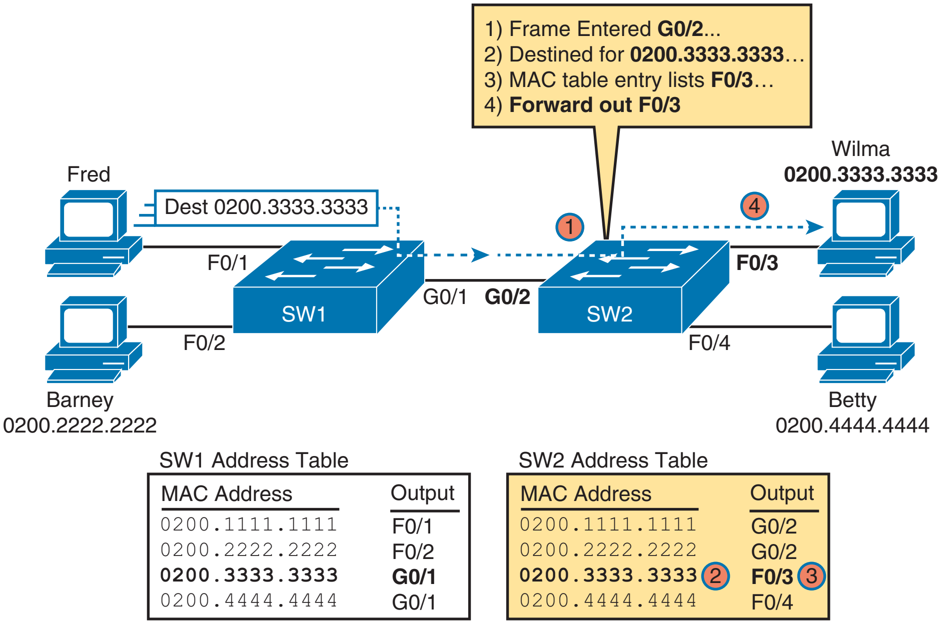
SW2 Address Table

MAC Address	Output
0200.1111.1111	G0/2
0200.2222.2222	G0/2
0200.3333.3333	**F0/3**
0200.4444.4444	F0/4

Figure 5-4 *Forwarding Decision with Two Switches: First Switch*

That same frame, after being forwarded by switch SW1, arrives at switch SW2, entering SW2's G0/2 interface. As shown in Figure 5-5, SW2 uses the same logic steps, but using SW2's table. The MAC table lists the forwarding instructions for that switch only. In this case, switch SW2 forwards the frame out its F0/3 port, based on SW2's MAC address table.

Key Topic



SW1 Address Table

MAC Address	Output
0200.1111.1111	F0/1
0200.2222.2222	F0/2
0200.3333.3333	**G0/1**
0200.4444.4444	G0/1

SW2 Address Table

MAC Address	Output
0200.1111.1111	G0/2
0200.2222.2222	G0/2
0200.3333.3333	**F0/3**
0200.4444.4444	F0/4

Figure 5-5 *Forwarding Decision with Two Switches: Second Switch*

NOTE The term *forward-versus-filter decision* emphasizes the switch's choice to forward the frame out some ports but not forward (filter) the frame from being sent out other ports.

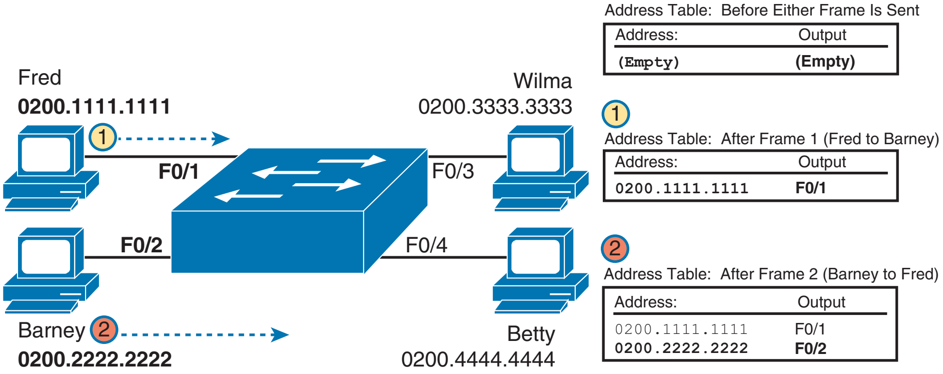
The examples so far use switches that happen to have a MAC table with all the MAC addresses listed. As a result, the destination MAC address in the frame is known to the switch. The frames are called **known unicast frames**, or simply known unicasts, because the destination address is a unicast address, and the destination is known. As shown in these examples, switches forward known unicast frames out one port: the port as listed in the MAC table entry for that MAC address.

Learning MAC Addresses

Thankfully, the networking staff does not have to type in all those MAC table entries. Instead, the switches do their second main function: to learn the MAC addresses and interfaces to put into its address table. With a complete MAC address table, the switch can make accurate forwarding and filtering decisions as just discussed.

Switches build the address table by listening to incoming frames and examining the *source MAC address* in the frame. If a frame enters the switch and the source MAC address is not in the MAC address table, the switch creates an entry in the table. That table entry lists the interface from which the frame arrived. Switch learning logic is that simple.

Figure 5-6 depicts the same single-switch topology network as Figure 5-3, but before the switch has built any address table entries. The figure shows the first two frames sent in this network: first a frame from Fred addressed to Barney, and then Barney's response, addressed to Fred.



Address Table: Before Either Frame Is Sent

Address:	Output
(Empty)	**(Empty)**

1 Address Table: After Frame 1 (Fred to Barney)

Address:	Output
0200.1111.1111	**F0/1**

2 Address Table: After Frame 2 (Barney to Fred)

Address:	Output
0200.1111.1111	F0/1
0200.2222.2222	**F0/2**

Figure 5-6 *Switch Learning: Empty Table and Adding Two Entries*

The figure does not show lines for where the frames flow, focusing instead on the arrival of the frames into the switch.

Focus on the learning process and how the MAC table grows at each step as shown on the right side of the figure. The switch begins with an empty MAC table, as shown in the upper-right part of the figure. Then Fred sends his first frame (labeled "1") to Barney, so the switch adds an entry for 0200.1111.1111, Fred's MAC address, associated with interface F0/1. Why F0/1? The frame sent by Fred entered the switch's F0/1 port. SW1's logic runs something like this: "The source is MAC 0200.1111.1111, the frame entered F0/1, so from my perspective, 0200.1111.1111 must be reachable out my port F0/1."

Continuing the example, when Barney replies in Step 2, the switch adds a second entry, this one for 0200.2222.2222, Barney's MAC address, along with interface F0/2. Why F0/2? The frame Barney sent entered the switch's F0/2 interface. Learning always occurs by looking at the source MAC address in the frame and adds the incoming interface as the associated port.

Flooding Unknown Unicast and Broadcast Frames

Now again turn your attention to the forwarding process, using the topology in Figure 5-5. What do you suppose the switch does with Fred's first frame, the one that occurred when there were no entries in the MAC address table? As it turns out, when there is no matching entry in the table, switches **forward** the frame out all interfaces (except the incoming interface) using a process called *flooding*. And the frame whose destination address is unknown to the switch is called an **unknown unicast frame**, or simply an *unknown unicast*.

Switches **flood** unknown unicast frames. Flooding means that the switch forwards copies of the frame out all ports, except the port on which the frame was received. The idea is simple: if you do not know where to send it, send it everywhere, to deliver the frame. And, by the way, that device will likely then send a reply—and then the switch can learn that device's MAC address and forward future frames out one port as a known unicast frame.

Switches also flood LAN **broadcast frames** (frames destined to the Ethernet broadcast address of FFFF.FFFF.FFFF) because this process helps deliver a copy of the frame to all devices in the LAN.

Figure 5-7 shows the same scenario as in Figure 5-6, with the first frame sent by Fred, when the switch's MAC table is empty—but focusing on switch forwarding. At Step 1, Fred sends the frame. At Step 2, the switch floods a copy of the frame out all three of the other interfaces.

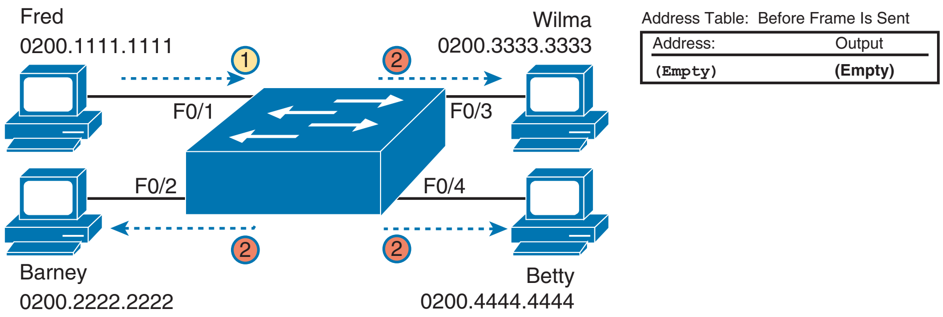


Figure 5-7 *Switch Flooding: Unknown Unicast Arrives, Floods Out Other Ports*

Avoiding Loops Using Spanning Tree Protocol

The third primary feature of LAN switches is loop prevention, as implemented by the **Spanning Tree Protocol (STP)**. Without STP, any flooded frames would loop for an indefinite period of time in Ethernet networks with physically redundant links. To prevent looping frames, STP blocks some ports from forwarding frames so that only one active path exists between any pair of LAN segments.

A simple example makes the need for STP more obvious. Remember, switches flood unknown unicast frames and broadcast frames. Figure 5-8 shows an unknown unicast frame, sent by Larry to Bob, which loops forever because the network has redundancy but no STP. Note that the figure shows one direction of the looping frame only, just to reduce clutter, but a copy of the frame would also loop the other direction.

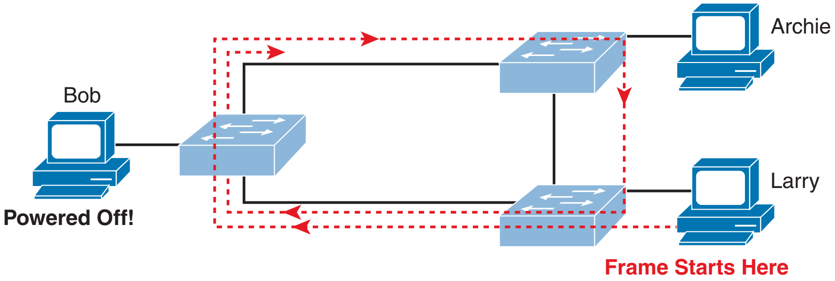


Figure 5-8 *Network with Redundant Links but Without STP: The Frame Loops Forever*

The flooding of this frame would cause the frame to rotate around the three switches; because none of the switches list Bob's MAC address in their address tables, each switch floods the frame. And while the flooding process is a good mechanism for forwarding unknown unicasts and broadcasts, the continual flooding of traffic frames as in the figure can completely congest the LAN to the point of making it unusable.

A topology like Figure 5-8, with redundant links, is good, but we need to prevent the bad effect of those looping frames. To avoid Layer 2 loops, all switches need to use STP. STP causes each interface on a switch to settle into either a blocking state or a forwarding state. *Blocking* means that the interface cannot forward or receive data frames, while *forwarding* means that the interface can send and receive data frames. If a correct subset of the interfaces is blocked, only a single currently active logical path exists between each pair of LANs.

Chapter 9 of this book, “Spanning Tree Protocol Concepts,” examines STP in depth, including how STP prevents loops.

LAN Switching Summary

Switches use Layer 2 logic, examining the Ethernet data-link header to choose how to process frames. In particular, switches make decisions to forward and filter frames, learn MAC addresses, and use STP to avoid loops, as follows:

Key Topic

Step 1. Switches forward frames based on the destination MAC address:

- **a.** If the destination MAC address is a broadcast, multicast, or unknown destination unicast (a unicast not listed in the MAC table), the switch floods the frame.
- **b.** If the destination MAC address is a known unicast address (a unicast address found in the MAC table):
  - i. If the outgoing interface listed in the MAC address table is different from the interface in which the frame was received, the switch forwards the frame out the outgoing interface.
  - ii. If the outgoing interface is the same as the interface in which the frame was received, the switch filters the frame, meaning that the switch simply ignores the frame and does not forward it.

Step 2. Switches learn MAC address table entries based on the source MAC address:

- **a.** For each received frame, note the source MAC address and incoming interface ID.
- **b.** If not yet in the MAC address table, add an entry listing the MAC address and incoming interface.

Step 3. Switches use STP to prevent loops by causing some interfaces to block, meaning that they do not send or receive frames.

Verifying and Analyzing Ethernet Switching

A Cisco Catalyst switch comes from the factory ready to switch frames. All you have to do is connect the power cable, plug in the Ethernet cables, and the switch starts switching incoming frames. Connect multiple switches together, and they are ready to forward frames between the switches as well. And the big reason behind this default behavior has to do with the default settings on the switches.

Cisco Catalyst switches come ready to get busy switching frames because of settings like these:

- The interfaces are enabled by default, ready to start working once a cable is connected.
- All interfaces are assigned to VLAN 1.
- 10/100 and 10/100/1000 interfaces use autonegotiation by default.

- The MAC learning, forwarding, flooding logic all works by default.
- STP is enabled by default.

This second section of the chapter examines how switches will work with these default settings, showing how to verify the Ethernet learning and forwarding process.

Demonstrating MAC Learning

To see a switch's MAC address table, use the **show mac address-table** command. With no additional parameters, this command lists all known MAC addresses in the MAC table, including some overhead static MAC addresses that you can ignore. To see all the dynamically learned MAC addresses only, instead use the **show mac address-table dynamic** command, as seen in Example 5-1.

Example 5-1 **show mac address-table dynamic** *for Figure 5-9*

```
SW1# show mac address-table dynamic
          Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0200.1111.1111    DYNAMIC     Fa0/1
   1    0200.2222.2222    DYNAMIC     Fa0/2
   1    0200.3333.3333    DYNAMIC     Fa0/3
   1    0200.4444.4444    DYNAMIC     Fa0/4
Total Mac Addresses for this criterion: 4
SW1#
```

First, focus on two columns of the table: the MAC Address and Ports columns of the table. The values should look familiar: they match the earlier single-switch example, as repeated here as Figure 5-9. Note the four MAC addresses listed, along with their matching ports, as shown in the figure.

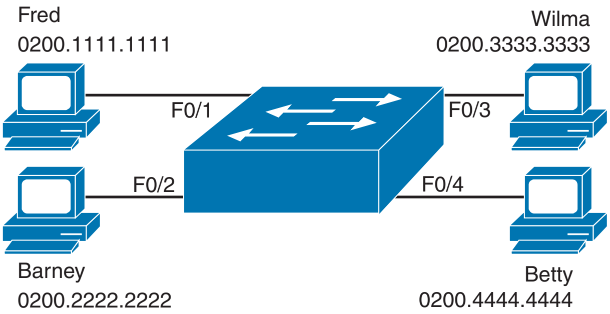


Figure 5-9 *Single Switch Topology Used in Verification Section*

Next, look at the Type field in the heading of the output table. The column tells us how the switch learned the MAC address as described earlier in this chapter; in this case, the switch learned all MAC addresses dynamically. You can also statically predefine MAC table entries using a couple of different features, including port security, and those would appear as Static in the Type column.

Finally, the VLAN column of the output gives us a chance to briefly discuss how virtual LANs (VLANs) impact switching logic. LAN switches forward Ethernet frames inside a VLAN. What that means is if a frame enters via a port in VLAN 1, then the switch will forward or flood that frame out other ports in VLAN 1 only, and not out any ports that happen to be assigned to another VLAN. Chapter 8, “Implementing Ethernet Virtual LANs,” looks at all the details of how switches forward frames when using VLANs.

Switch Interfaces

The first example assumes that you installed the switch and cabling correctly, and that the switch interfaces work. You can easily check the status of those interfaces with the **show interfaces status** command, as shown in Example 5-2.

Example 5-2 **show interfaces status** *on Switch SW1*

```
SW1# show interfaces status

          Name                           Vlan       Duplex  Speed  Type
Fa0/1                        connected   1          a-full  a-100 10/100BaseTX
Fa0/2                        connected   1          a-full  a-100 10/100BaseTX
Fa0/3                        connected   1          a-full  a-100 10/100BaseTX
Fa0/4                        connected   1          a-full  a-100 10/100BaseTX
Fa0/5                        notconnect  1          auto     auto 10/100BaseTX
Fa0/6                        notconnect  1          auto     auto 10/100BaseTX
Fa0/7                        notconnect  1          auto     auto 10/100BaseTX
Fa0/8                        notconnect  1          auto     auto 10/100BaseTX
Fa0/9                        notconnect  1          auto     auto 10/100BaseTX
Fa0/10                       notconnect  1          auto     auto 10/100BaseTX
Fa0/11                       notconnect  1          auto     auto 10/100BaseTX
Fa0/12                       notconnect  1          auto     auto 10/100BaseTX
Fa0/13                       notconnect  1          auto     auto 10/100BaseTX
Fa0/14                       notconnect  1          auto     auto 10/100BaseTX
Fa0/15                       notconnect  1          auto     auto 10/100BaseTX
Fa0/16                       notconnect  1          auto     auto 10/100BaseTX
Fa0/17                       notconnect  1          auto     auto 10/100BaseTX
Fa0/18                       notconnect  1          auto     auto 10/100BaseTX
Fa0/19                       notconnect  1          auto     auto 10/100BaseTX
Fa0/20                       notconnect  1          auto     auto 10/100BaseTX
Fa0/21                       notconnect  1          auto     auto 10/100BaseTX
Fa0/22                       notconnect  1          auto     auto 10/100BaseTX
Fa0/23                       notconnect  1          auto     auto 10/100BaseTX
Fa0/24                       notconnect  1          auto     auto 10/100BaseTX
Gi0/1                        notconnect  1          auto     auto 10/100/1000BaseTX
Gi0/2                        notconnect  1          auto     auto 10/100/1000BaseTX
```

Focus on the Port column for a moment. As a reminder, Cisco Catalyst switches name their ports based on the fastest specification supported, so in this case, the switch has 24 interfaces named FastEthernet and two named GigabitEthernet. Many commands abbreviate those terms, this time as Fa for FastEthernet and Gi for GigabitEthernet. (The example happens to come from a Cisco Catalyst switch that has 24 10/100 ports and two 10/100/1000 ports.)

The Status column, of course, tells us the status or state of the port. In this case, the lab switch had cables and devices connected to ports F0/1–F0/4 only, with no other cables connected. As a result, those first four ports have a state of connected, meaning that the ports have a cable and are functional. The notconnect state means that the port is not yet functioning. It may mean that there is no cable installed, but other problems may exist as well. (The section “Analyzing Switch Interface Status and Statistics,” in Chapter 7, “Configuring and Verifying Switch Interfaces,” works through the details of what causes a switch interface to fail.)

NOTE You can see the status for a single interface in a couple of ways. For instance, for F0/1, the command **show interfaces f0/1 status** lists the status in a single line of output as in Example 5-2. The **show interfaces f0/1** command (without the **status** keyword) displays a detailed set of messages about the interface.

The **show interfaces** command has a large number of options. One particular option, the **counters** option, lists statistics about incoming and outgoing frames on the interfaces. In particular, it lists the number of unicast, multicast, and broadcast frames (both the in and out directions), and a total byte count for those frames. Example 5-3 shows an example, again for interface F0/1.

Example 5-3 **show interfaces f0/1 counters** *on Switch SW1*

```
SW1# show interfaces f0/1 counters

Port            InOctets    InUcastPkts      InMcastPkts      InBcastPkts
Fa0/1            1223303          10264              107               18

Port           OutOctets    OutUcastPkts    OutMcastPkts     OutBcastPkts
Fa0/1            3235055           13886           22940              437
```

Finding Entries in the MAC Address Table

With a single switch and only four hosts connected to them, you can just read the details of the MAC address table and find the information you want to see. However, in real networks, with lots of interconnected hosts and switches, just reading the output to find one MAC address can be hard to do. You might have hundreds of entries—page after page of output—with each MAC address looking like a random string of hex characters. (The book uses easy-to-recognize MAC addresses to make it easier to learn.)

Thankfully, Cisco IOS supplies several more options on the **show mac address-table** command to make it easier to find individual entries. First, if you know the MAC address, you can search for it—just type in the MAC address at the end of the command, as shown in Example 5-4. All you have to do is include the **address** keyword, followed by the actual MAC address. If the address exists, the output lists the address. Note that the output lists the exact same information in the exact same format, but it lists only the line for the matching MAC address.

Example 5-4 **show mac address-table dynamic** *with the* **address** *Keyword*

```
SW1# show mac address-table dynamic address 0200.1111.1111
          Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0200.1111.1111    DYNAMIC     Fa0/1
Total Mac Addresses for this criterion: 1
```

While this information is useful, often the engineer troubleshooting a problem does not know the MAC addresses of the devices connected to the network. Instead, you might be troubleshooting while looking at a network topology diagram and want to look at all the MAC addresses learned off a particular port. IOS supplies that option with the **show mac address-table dynamic interface** command. Example 5-5 shows one example, for switch SW1's F0/1 interface.

Example 5-5 **show mac address-table dynamic** *with the* **interface** *Keyword*

```
SW1# show mac address-table dynamic interface fastEthernet 0/1
          Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0200.1111.1111    DYNAMIC     Fa0/1
Total   Mac Addresses for this criterion: 1
```

Finally, you may also want to find the MAC address table entries for one VLAN. You guessed it—you can add the **vlan** parameter, followed by the VLAN number. Example 5-6 shows two such examples from the same switch SW1 from Figure 5-9—one for VLAN 1, where all four devices reside, and one for a nonexistent VLAN 2.

Example 5-6 *The* **show mac address-table vlan** *command*

```
SW1# show mac address-table dynamic vlan 1
          Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0200.1111.1111    DYNAMIC     Fa0/1
   1     0200.2222.2222    DYNAMIC     Fa0/2
   1     0200.3333.3333    DYNAMIC     Fa0/3
   1     0200.4444.4444    DYNAMIC     Fa0/4
Total Mac Addresses for this criterion: 4
SW1#
SW1# show mac address-table dynamic vlan 2
          Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
SW1#
```

Managing the MAC Address Table (Aging, Clearing)

This chapter closes with a few comments about how switches manage their MAC address tables. MAC addresses do not remain in the table indefinitely. The switch will remove the entries due to age, due to the table filling, and you can remove entries using a command.

First, for aging out MAC table entries, switches remove entries that have not been used for a defined number of seconds (default of 300 seconds). To do that, switches keep a timer for each MAC table entry that increases over time. However, the switch resets the timer to 0 when it receives another frame with that same source MAC address. Timers that have not been reset continue to grow, and once a timer reaches the aging setting, the switch removes the MAC table entry.

Example 5-7 shows the aging timer setting for the entire switch. The aging time can be configured to a different time, globally and per-VLAN using the **mac address-table aging-time** *time-in-seconds* [**vlan** *vlan-number*] global configuration command. The example shows a case with all defaults, with the global setting of 300 seconds, and no per-VLAN overrides.

Example 5-7 *The MAC Address Default Aging Timer Displayed*

```
SW1# show mac address-table aging-time
Global Aging Time:  300
Vlan    Aging Time
----    ----------
SW1#

SW1# show mac address-table count

Mac Entries for Vlan 1:
---------------------------
Dynamic Address Count  : 4
Static  Address Count  : 0
Total Mac Addresses    : 4

Total Mac Address Space Available: 7299
```

Each switch also removes the oldest table entries, even if they are younger than the aging time setting, if the table fills. The MAC address table uses content-addressable memory (CAM), a physical memory that has great table lookup capabilities. However, the size of the table depends on the size of the CAM in a particular model of switch and based on some configurable settings in the switch. When a switch tries to add a new MAC table entry and finds the table full, the switch removes the oldest table entry to make space. For perspective, the end of Example 5-7 lists the size of a Cisco Catalyst switch's MAC table at about 8000 entries—the same four existing entries from the earlier examples, with space for 7299 more.

Finally, you can remove the dynamic entries from the MAC address table with the **clear mac address-table dynamic** command. Note that the **show** commands in this chapter can be executed from user and enable mode, but the **clear** command happens to be an enable mode command. The command also allows parameters to limit the types of entries cleared, as follows:

- **By VLAN: clear mac address-table dynamic vlan** *vlan-number*
- **By Interface: clear mac address-table dynamic interface** *interface-id*
- **By MAC address: clear mac address-table dynamic address** *mac-address*

MAC Address Tables with Multiple Switches

Finally, to complete the discussion, it helps to think about an example with multiple switches, just to emphasize how MAC learning, forwarding, and flooding happen independently on each LAN switch.

Consider the topology in Figure 5-10, and pay close attention to the port numbers. The ports were purposefully chosen so that neither switch used any of the same ports for this example. That is, switch SW2 does have a port F0/1 and F0/2, but I did not plug any devices into those ports when making this example. Also note that all ports are in VLAN 1, and as with the other examples in this chapter, all default configuration is used other than the hostname on the switches.

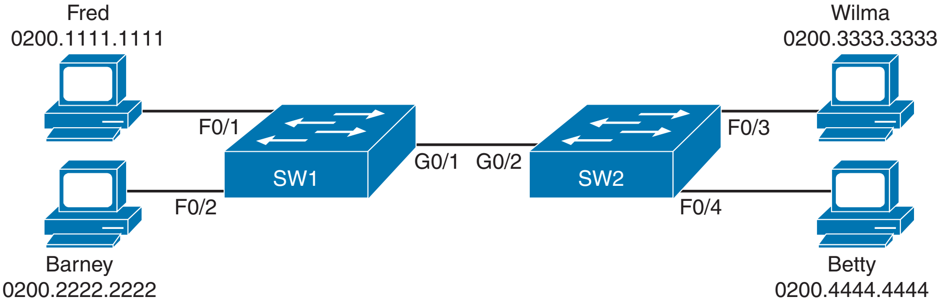


Figure 5-10 *Two-Switch Topology Example*

Think about a case in which both switches learn all four MAC addresses. For instance, that would happen if the hosts on the left communicate with the hosts on the right. SW1's MAC address table would list SW1's own port numbers (F0/1, F0/2, and G0/1) because SW1 uses that information to decide where SW1 should forward frames. Similarly, SW2's MAC table lists SW2's port numbers (F0/3, F0/4, G0/2 in this example). Example 5-8 shows the MAC address tables on both switches for that scenario.

Example 5-8 *The MAC Address Table on Two Switches*

```
SW1# show mac address-table dynamic
          Mac Address Table
-------------------------------------------


Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
   1    0200.1111.1111    DYNAMIC     Fa0/1
   1    0200.2222.2222    DYNAMIC     Fa0/2
   1    0200.3333.3333    DYNAMIC     Gi0/1
   1    0200.4444.4444    DYNAMIC     Gi0/1
Total Mac Addresses for this criterion: 4
```

```
! The next output is from switch SW2
SW2# show mac address-table dynamic
   1    0200.1111.1111    DYNAMIC     Gi0/2
   1    0200.2222.2222    DYNAMIC     Gi0/2
   1    0200.3333.3333    DYNAMIC     Fa0/3
   1    0200.4444.4444    DYNAMIC     Fa0/4
Total Mac Addresses for this criterion: 4
```

Chapter Review

Review this chapter's material using either the tools in the book or interactive tools for the same material found on the book's companion website. Table 5-2 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 5-2 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP
Do labs		Book, Sim Lite, blog
Review command tables		Book
Watch video		Website

Review All the Key Topics

Key Topic

Table 5-3 Key Topics for Chapter 5

Key Topic Element	Description	Page Number
List	Three main functions of a LAN switch	115
Figure 5-3	Process to forward a known unicast frame	116
Figure 5-5	Process to forward a known unicast, second switch	118
Figure 5-6	Process to learn MAC addresses	119
List	Summary of switch forwarding logic	121
Example 5-1	The **show mac address-table dynamic** command	122

Do Labs

The Sim Lite software is a version of Pearson's full simulator learning product with a subset of the labs, included free with this book. The subset of labs mostly relates to this part of the book, so take the time to try some of the labs.

As always, also check the author's blog site pages for configuration exercises (Config Labs) at http://www.certskills.com.

Key Terms You Should Know

broadcast frame, flood, forward, known unicast frame, MAC address table, Spanning Tree Protocol (STP), unknown unicast frame

Command References

Table 5-4 lists the verification commands used in this chapter. As an easy review exercise, cover the left column, read the right, and try to recall the command without looking. Then repeat the exercise, covering the right column, and try to recall what the command does.

Table 5-4 Chapter 5 EXEC Command Reference

Command	Mode/Purpose/Description
show mac address-table	Shows all MAC table entries of all types
show mac address-table dynamic	Shows all dynamically learned MAC table entries
show mac address-table dynamic vlan *vlan-id*	Shows all dynamically learned MAC table entries in that VLAN
show mac address-table dynamic address *mac-address*	Shows the dynamically learned MAC table entries with that MAC address
show mac address-table dynamic interface *interface-id*	Shows all dynamically learned MAC table entries associated with that interface
show mac address-table count	Shows the number of entries in the MAC table and the total number of remaining empty slots in the MAC table

Command	Mode/Purpose/Description
show mac address-table aging-time	Shows the global and per-VLAN aging timeout for inactive MAC table entries
show interfaces *id* **counters**	Lists packet counters for the listed interface ID
show interfaces status	Lists one line per interface on the switch, with basic status and operating/ information for each
clear mac address-table dynamic [vlan *vlan-number*] **[interface** *interface-id*] **[address** *mac-address*]	Clears (removes) dynamic MAC table entries: either all (with no parameters), or a subset based on VLAN ID, interface ID, or a specific MAC address

Note that this chapter also includes reference to one configuration command, so it does not call for the use of a separate table. For review, the command is

mac address-table aging-time *time-in-seconds* [**vlan** *vlan-number*]

This page intentionally left blank

CHAPTER 6

Configuring Basic Switch Management

This chapter covers the following exam topics:

- **1.0 Network Fundamentals**
  - **1.6 Configure and verify IPv4 addressing and subnetting**
- **2.0 Network Access**
  - **2.8 Describe network device management access (Telnet, SSH, HTTP, HTTPS, console, RADIUS/TACACS+)**
- **4.0 IP Services**
  - **4.6 Configure and verify DHCP client and relay**
  - **4.8 Configure network devices for remote access using SSH**
- **5.0 Security Fundamentals**
  - **5.3 Configure device access control using local passwords**

The tasks of a switch fall into a small set of categories called planes. The data plane includes the process of forwarding frames received by the switch. The control plane refers to the processes that control and change the switch's data plane. The control plane includes configuration to enable or disable an interface, to control the speed used by each interface, and the dynamic processes of Spanning Tree to block some ports to prevent loops, and so on. The third plane, the management plane, refers to device management features. Those include Telnet and SSH, used to connect to the CLI, and other management features.

This chapter discusses the most basic management plane features in a Cisco switch. The first section of the chapter discusses configuring different kinds of login security for console, Telnet, and SSH users. The second section shows how to enable remote switch management by configuring switch IPv4 settings. The last section then explains a few practical matters that can make your life in the lab a little easier.

"Do I Know This Already?" Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book as well as on the companion website, includes both the answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 6-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Securing the Switch CLI	1–3
Enabling IPv4 for Remote Access	4–5
Miscellaneous Settings Useful in the Lab	6

1. Imagine that you have configured the **enable secret** command, followed by the **enable password** command, from the console. You log out of the switch and log back in at the console. Which command defines the password that you had to enter to access privileged mode?
   a. **enable password**
   b. **enable secret**
   c. Neither
   d. The **password** command, if it is configured
2. An engineer wants to set up simple password protection with no usernames for some switches in a lab, for the purpose of keeping curious coworkers from logging in to the lab switches from their desktop PCs. Which of the following commands would be a useful part of that configuration?
   a. A **login** vty mode subcommand
   b. A **password** *password* console subcommand
   c. A **login local** vty subcommand
   d. A **transport input ssh** vty subcommand
3. An engineer had formerly configured a Cisco 2960 switch to allow Telnet access so that the switch expected a password of **mypassword** from the Telnet user. The engineer then changed the configuration to support Secure Shell. Which of the following commands could have been part of the new configuration? (Choose two answers.)
   a. A **username** *name* **secret** *password* vty mode subcommand
   b. A **username** *name* **secret** *password* global configuration command
   c. A **login local** vty mode subcommand
   d. A **transport input ssh** global configuration command
4. An engineer's desktop PC connects to a switch at the main site. A router at the main site connects to each branch office through a serial link, with one small router and switch at each branch. Which of the following commands must be configured on the branch office switches, in the listed configuration mode, to allow the engineer to telnet to the branch office switches and supply only a password to log in? (Choose three answers.)
   a. The **ip address** command in interface configuration mode
   b. The **ip address** command in global configuration mode
   c. The **ip default-gateway** command in VLAN configuration mode
   d. The **ip default-gateway** command in global configuration mode
   e. The **password** command in console line configuration mode
   f. The **password** command in vty line configuration mode

5. A Layer 2 switch configuration places all its physical ports into VLAN 2. An attached router uses address/mask 172.16.2.254/24. The IP address plan calls for the switch to use address/mask 172.16.2.250/24 and to use the router as its default gateway. The switch needs to support SSH connections into the switch from any subnet in the network. Which of the following commands are part of the required configuration in this case? (Choose two answers.)
   a. The **ip address 172.16.2.250 255.255.255.0** command in interface vlan 1 configuration mode.
   b. The **ip address 172.16.2.250 255.255.255.0** command in interface vlan 2 configuration mode.
   c. The **ip default-gateway 172.16.2.254** command in global configuration mode.
   d. The switch cannot support SSH because all its ports connect to VLAN 2, and the IP address must be configured on interface VLAN 1.
6. Which of the following line subcommands tells a switch to wait until a show command's output has completed before displaying log messages on the screen?
   a. **logging synchronous**
   b. **no ip domain-lookup**
   c. **exec-timeout 0 0**
   d. **history size 15**

Foundation Topics

Securing the Switch CLI

By default, a user can connect to the console and reach **enable mode** with no security checks and no passwords required. In contrast, the default settings disallow all **Telnet** and **Secure Shell (SSH)** users from even seeing a login prompt. Those defaults make sense, given that if you can get to the console port of the switch, you already have control over the switch physically. But clearly, protecting the console makes sense, as does opening up SSH and Telnet access to appropriate users.

This first topic in the chapter examines how to configure login security for a Cisco Catalyst switch. Securing the CLI includes protecting access to enable mode, because from enable mode, an attacker could reload the switch or change the configuration. Protecting user mode is also important, because attackers can see the status of the switch, learn about the network, and find new ways to attack the network.

In particular, this section covers the following login security topics:

- Securing user mode and privileged mode with simple passwords
- Securing user mode access with local usernames
- Securing user mode access with external authentication servers
- Securing remote access with Secure Shell (SSH)

Note that all remote management protocols, like Telnet and SSH, require IP configuration on the switch, which is not discussed until the second major section of this chapter, "Enabling IPv4 for Remote Access."

Securing User Mode and Privileged Mode with Simple Passwords

The defaults work great for a brand new switch, but in production, you will want to secure access through the console as well as enable remote login via Telnet and/or SSH so you can sit at your desk and log in to all the switches in the LAN. Keep in mind, however, that you should not open the switch for just anyone to log in and change the configuration, so some type of secure login should be used.

Most people use a simple shared password for access to lab gear. This method uses a password only—with no username—with one password for console users and a different password for Telnet users. Console users must supply the *console password*, as configured in console line configuration mode. Telnet users must supply the *Telnet password*, also called the vty password, so called because the configuration sits in vty line configuration mode. Figure 6-1 summarizes these options for using shared passwords from the perspective of the user logging in to the switch.

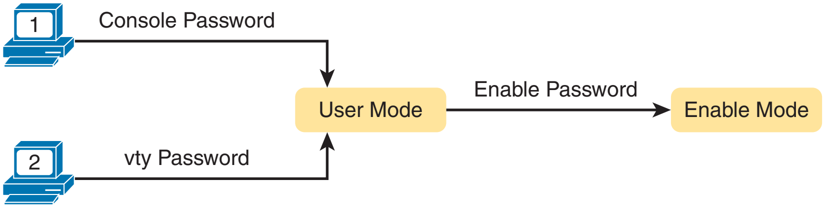


Figure 6-1 *Simple Password Security Concepts*

NOTE This section refers to several passwords as *shared* passwords. Users share these passwords in that all users must know and use that same password. In other words, each user does not have a unique username/password to use, but rather, all the appropriate staff knows and uses the same password.

In addition, Cisco switches protect enable mode (also called privileged mode) with yet another shared password called the *enable password*. From the perspective of the network engineer connecting to the CLI of the switch, once in user mode, the user types the **enable** EXEC command. This command prompts the user for this enable password; if the user types the correct password, IOS moves the user to enable mode.

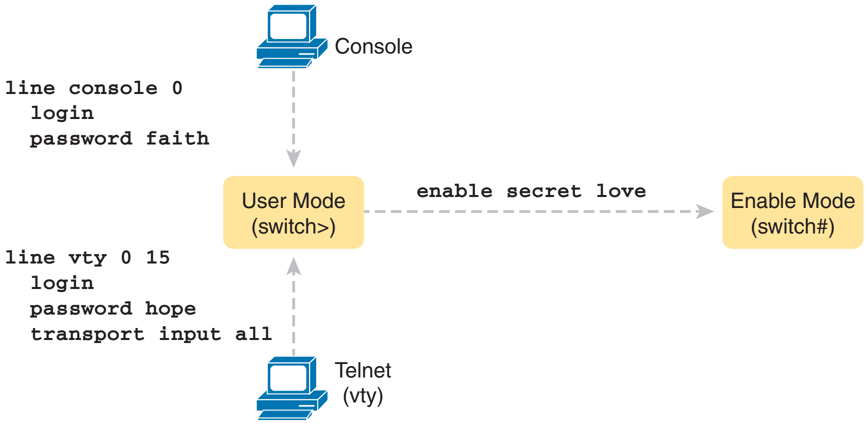
Example 6-1 shows an example of the user experience of logging in to a switch from the console when the shared console password and the shared enable password have both been set. Note that before this example began, the user started the terminal emulator, physically connected a laptop to the console cable, and then pressed the Enter key to make the switch respond as shown at the top of the example.

Example 6-1 *Console Login and Movement to Enable Mode*

```
(User now presses enter to start the process. This line of text does not appear.)

User Access Verification

Password: faith
Switch> enable
Password: love
Switch#
```

Note that the example shows the password text as if typed (faith and love), along with the **enable** command that moves the user from user mode to enable mode. In reality, the switch hides the passwords when typed, to prevent someone from reading over your shoulder to see the passwords.

To configure the shared passwords for the console, Telnet, and for enable mode, you need to configure several commands. However, the parameters of the commands can be pretty intuitive. Figure 6-2 shows the configuration of all three of these passwords.


```
line console 0
  login
  password faith
```

```
enable secret love
```

```
line vty 0 15
  login
  password hope
  transport input all
```

Figure 6-2 *Simple Password Security Configuration*

The configuration for these three passwords does not require a lot of work. First, the console and vty password configuration sets the password based on the context: console mode for the console (**line con 0**), and vty line configuration mode for the Telnet password (**line vty 0 15**). Then inside console mode and vty mode, respectively, the two commands in each mode are as follows:

password *password-value*: Defines the actual password used on the console or vty

login: Tells IOS to enable the use of a simple shared password (with no username) on this line (console or vty), so that the switch asks the user for a password

Answers to the "Do I Know This Already?" quiz:

1 B **2** A **3** B, C **4** A, D, F **5** B, C **6** A

The configured enable password, shown on the right side of the figure, applies to all users, no matter whether they connect to user mode via the console, Telnet, or otherwise. The command to configure the enable password is a global configuration command: **enable secret** *password-value*.

NOTE Older IOS versions used the command **enable password** *password-value* to set the enable password, and that command still exists in IOS. However, the **enable secret** command is much more secure. In real networks, use **enable secret**. Chapter 10, "Securing Network Devices," in the *CCNA 200-301 Official Cert Guide*, *Volume 2*, Second Edition, explains more about the security levels of various password mechanisms, including a comparison of the **enable secret** and **enable password** commands.

To help you follow the process, and for easier study later, use the configuration checklist before the example. The configuration checklist collects the required and optional steps to configure a feature as described in this book. The configuration checklist for shared passwords for the console, Telnet, and enable passwords is

Config Checklist

Step 1. Configure the enable password with the **enable secret** *password-value* command.

Step 2. Configure the console password:

- **a.** Use the **line con 0** command to enter console configuration mode.
- **b.** Use the **password** *password-value* subcommand to set the value of the console password.
- **c.** Use the **login** subcommand to enable console password security using a simple password.

Step 3. Configure the Telnet (vty) password:

- **a.** Use the **line vty 0 15** command to enter vty configuration mode for all 16 vty lines (numbered 0 through 15).
- **b.** Use the **password** *password-value* subcommand to set the value of the vty password.
- **c.** Use the **login** subcommand to enable console password security using a simple password.
- **d.** Use the **transport input all** subcommand (or similar) to enable Telnet as an input protocol for the vty lines.

NOTE The section "Securing Remote Access with Secure Shell," later in this chapter, provides more detail about the **transport input** subcommand.

Example 6-2 shows the configuration process as noted in the configuration checklist, along with setting the enable secret password. Note that the lines which begin with a **!** are comment lines; they are there to guide you through the configuration.

Example 6-3 shows the resulting configuration in the switch per the **show running-config** command. The gray lines highlight the new configuration. Note that many unrelated lines of output have been deleted from the output to keep focused on the password configuration.

Key Topic

Example 6-2 *Configuring Basic Passwords*

```
! Enter global configuration mode and set the enable password.
!
Switch# configure terminal
Switch(config)# enable secret love
!
! At Step 2 in the checklist, enter console configuration mode, set the
! password value to “faith” and enable simple passwords for the console.
! The exit command moves the user back to global config mode.
!
Switch#(config)# line console 0
Switch#(config-line)# password faith
Switch#(config-line)# login
Switch#(config-line)# exit
!
! The next few lines do basically the same configuration, except it is
! for the vty lines. Telnet users will use "hope" to login.
!
Switch#(config)# line vty 0 15
Switch#(config-line)# password hope
Switch#(config-line)# login
Switch#(config-line)# transport input all
Switch#(config-line)# end
Switch#
```

Example 6-3 *Resulting Running-Config File (Subset) per Example 6-2 Configuration*

```
Switch# show running-config
!
Building configuration...

Current configuration: 1333 bytes
!
version 12.2
!
enable secret 5 $1$OwtI$A58c2XgqWyDNeDnv51mNR.
!
interface FastEthernet0/1
!
interface FastEthernet0/2
!
! Several lines have been omitted here - in particular, lines for
! FastEthernet interfaces 0/3 through 0/23.
!
interface FastEthernet0/24
```

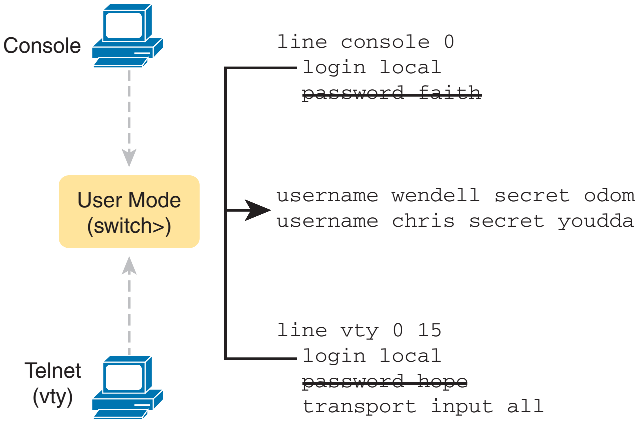
```
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/2
!
line con 0
 password faith
login
!
line vty 0 4
 password hope
 login
 transport input all
!
line vty 5 15
 password hope
 login
 transport input all
```

NOTE For historical reasons, the output of the **show running-config** command, in the last six lines of Example 6-3, separates the first five vty lines (0 through 4) from the rest (5 through 15).

Securing User Mode Access with Local Usernames and Passwords

Cisco switches support two other login security methods that both use per-user username/password pairs instead of a shared password with no username. One method, referred to as **local usernames** and passwords, configures the username/password pairs locally—that is, in the switch's configuration. Switches support this local username/password option for the console, for Telnet, and even for SSH, but do not replace the enable password used to reach enable mode.

The configuration to migrate from using the simple shared passwords to instead using local usernames/passwords requires only some small configuration changes, as shown in Figure 6-3.



```
line console 0
  login local
  ~~password faith~~

username wendell secret odom
username chris secret youdda

line vty 0 15
  login local
  ~~password hope~~
  transport input all
```

Figure 6-3 *Configuring Switches to Use Local Username Login Authentication*

Working through the configuration in the figure, first, the switch of course needs to know the list of username/password pairs. To create these, repeatedly use the **username** *name* **secret** *password* global configuration command. Then, to enable this different type of console or Telnet security, simply enable this login security method with the **login local** line. Basically, this command means "use the local list of usernames for login." You can also use the **no password** command (without even typing in the password) to clean up any remaining password subcommands from console or vty mode because these commands are not needed when using local usernames and passwords.

The following checklist details the commands to configure local username login, mainly as a method for easier study and review:

Config Checklist

Step 1. Use the **username** *name* **secret** *password* global configuration command to add one or more username/password pairs on the local switch.

Step 2. Configure the console to use locally configured username/password pairs:

- **a.** Use the **line con 0** command to enter console configuration mode.
- **b.** Use the **login local** subcommand to enable the console to prompt for both username and password, checked versus the list of local usernames/passwords.
- **c.** (Optional) Use the **no password** subcommand to remove any existing simple shared passwords, just for good housekeeping of the configuration file.

Step 3. Configure Telnet (vty) to use locally configured username/password pairs.

- **a.** Use the **line vty 0 15** command to enter vty configuration mode for all 16 vty lines (numbered 0 through 15).
- **b.** Use the **login local** subcommand to enable the switch to prompt for both username and password for all inbound Telnet users, checked versus the list of local usernames/passwords.
- **c.** (Optional) Use the **no password** subcommand to remove any existing simple shared passwords, just for good housekeeping of the configuration file.
- **d.** Use the **transport input all** subcommand (or similar) to enable Telnet as an input protocol for the vty lines.

When a Telnet user connects to the switch configured as shown in Figure 6-3, the user will be prompted first for a username and then for a password, as shown in Example 6-4. The username/password pair must be from the list of local usernames; otherwise, the login is rejected.

Example 6-4 *Telnet Login Process After Applying Configuration in Figure 6-3*

```
SW2# telnet 10.9.9.19
Trying 10.9.9.19 ... Open



User Access Verification

```

```
Username: wendell
Password:
SW1> enable
Password:
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)#^Z
SW1#
*Mar 1 02:00:56.229: %SYS-5-CONFIG_I: Configured from console by wendell on vty0
(10.9.9.19)
```

NOTE Example 6-4 does not show the password value as having been typed because Cisco switches do not display the typed password for security reasons.

Securing User Mode Access with External Authentication Servers

The end of Example 6-4 points out one of the many security improvements when requiring each user to log in with their own username. The end of the example shows the user entering configuration mode (**configure terminal**) and then immediately leaving (**end**). Note that when a user exits configuration mode, the switch generates a **log message**. If the user logged in with a username, the log message identifies that username; note the "wendell" in the log message.

However, using a username/password configured directly on the switch causes some administrative headaches. For instance, every switch and router needs the configuration for all users who might need to log in to the devices. Then, when any changes need to happen, like an occasional change to the passwords for good security practices, the configuration of all devices must be changed.

A better option would be to use tools like those used for many other IT login functions. Those tools allow for a central place to securely store all username/password pairs, with tools to make users change their passwords regularly, tools to revoke users when they leave their current jobs, and so on.

Cisco switches allow exactly that option using an external server called an authentication, authorization, and accounting (**AAA**) server. These servers hold the usernames/passwords. Typically, these servers allow users to do self-service and forced maintenance to their passwords. Many production networks use AAA servers for their switches and routers today.

The underlying login process requires some additional work on the part of the switch for each user login, but once set up, the username/password administration is much less. When an **AAA server** is used for authentication, the switch (or router) simply sends a message to the AAA server asking whether the username and password are allowed, and the AAA server replies. Figure 6-4 shows an example, with the user first supplying a username/password, the switch asking the AAA server, and the server replying to the switch stating that the username/password is valid.

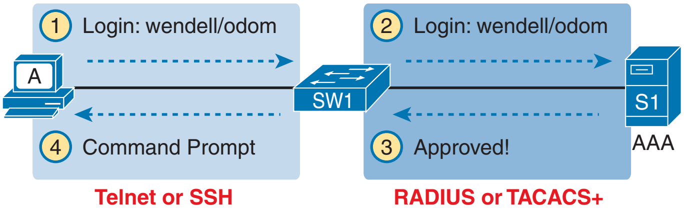


Figure 6-4 *Basic Authentication Process with an External AAA Server*

While the figure shows the general idea, note that the information flows with a couple of different protocols. On the left, the connection between the user and the switch or router uses Telnet or SSH. On the right, the switch and AAA server typically use either the RADIUS or TACACS+ protocol, both of which encrypt the passwords as they traverse the network.

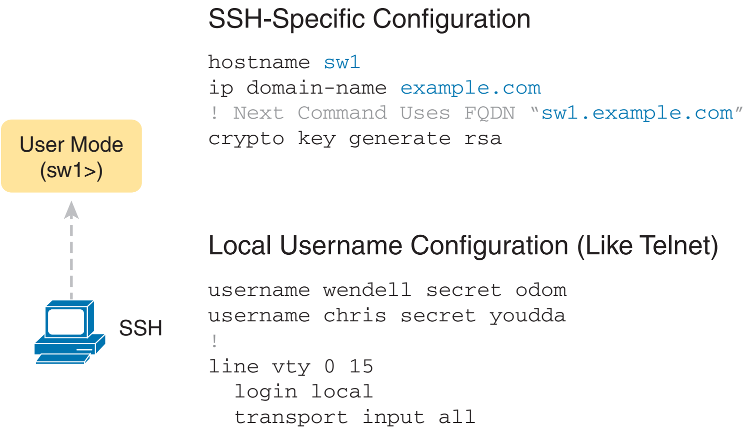
Securing Remote Access with Secure Shell

So far, this chapter has focused on the console and on Telnet, mostly ignoring SSH. Telnet has one serious disadvantage: all data in the Telnet session flows as clear text, including the password exchanges. So, anyone who can capture the messages between the user and the switch (in what is called a man-in-the-middle attack) can see the passwords. SSH encrypts all data transmitted between the SSH client and server, protecting the data and passwords.

SSH can use the same local login authentication method as Telnet, with the locally configured username and password. (SSH cannot rely on authentication methods that do not include a username, like shared passwords.) So, the configuration to support local usernames for Telnet, as shown previously in Figure 6-3, also enables local username authentication for incoming SSH connections.

Figure 6-5 shows one example configuration of what is required to support SSH. The figure repeats the local username configuration as shown earlier in Figure 6-3, as used for Telnet. Figure 6-5 shows three additional commands required to complete the configuration of SSH on the switch.

Key Topic



SSH-Specific Configuration

```
hostname sw1
ip domain-name example.com
! Next Command Uses FQDN "sw1.example.com"
crypto key generate rsa
```

Local Username Configuration (Like Telnet)

```
username wendell secret odom
username chris secret youdda
!
line vty 0 15
  login local
  transport input all
```

Figure 6-5 *Adding SSH Configuration to Local Username Configuration*

IOS uses the three SSH-specific configuration commands in the figure to create the SSH encryption keys. The SSH server uses the fully qualified domain name (FQDN) of the switch as input to create that key. The switch creates the FQDN from the hostname and domain name of the switch. Figure 6-5 begins by setting both values (just in case they are not already configured). Then the third command, the **crypto key generate rsa** command, generates the SSH encryption keys.

Seeing the configuration happen in configuration mode, step by step, can be particularly helpful with SSH configuration. Note in particular that in this example, the **crypto key** command prompts the user for the key modulus; you could also add the parameters **modulus** *modulus-value* to the end of the **crypto key** command to add this setting on the command. Example 6-5 shows the commands in Figure 6-5 being configured, with the encryption key as the final step.

Example 6-5 *SSH Configuration Process to Match Figure 6-5*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
!
! Step 1 next. The hostname is already set, but it is repeated just
! to be obvious about the steps.
!
SW1(config)# hostname SW1
SW1(config)# ip domain name example.com
SW1(config)# crypto key generate rsa
The name for the keys will be: SW1.example.com

Choose the size of the key modulus in the range of 512 to 2048 for your
  General Purpose Keys. Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [1024]: 1024
% Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 4 seconds)
SW1(config)#
!
! Optionally, set the SSH version to version 2 (only) - preferred
!
SW1(config)# ip ssh version 2
!
! Next, configure the vty lines for local username support, just like
! with Telnet
!
SW1(config)# line vty 0 15
SW1(config-line)# login local
SW1(config-line)# transport input all
SW1(config-line)# exit
!
```

```
! Define the local usernames, just like with Telnet
!
SW1(config)# username wendell secret odom
SW1(config)# username chris secret youdaman
SW1(config)# ^Z
SW1#
```

NOTE Older IOS versions used the syntax **ip domain-name** *domain-name* rather than the newer **ip domain name** *domain-name* (with a space instead of a dash.)

Both the Telnet and SSH examples throughout this chapter so far list the **transport input all** subcommand in vty configuration mode. The **transport input** command identifies the protocols allowed in the vty ports, with the **all** keyword including SSH and Telnet. Using **transport input all** lets you support everything when getting started learning, but for production devices you might want to instead choose some different options, like not supporting Telnet at all due to its poor security. Some common options include

transport input all or **transport input telnet ssh:** Support both Telnet and SSH

transport input none: Support neither

transport input telnet: Support only Telnet

transport input ssh: Support only SSH

Over the years, the default settings for this command have varied a bit based on device type, OS, and OS version. As a result, for production devices, it makes sense to pick the setting you want and configure it, rather than relying on your memory of a default setting for a particular device and software version. For instance, many companies prefer to disable any possibility of Telnet access, but allow SSH, using **transport input ssh.**

For the exams, be ready to look for the **transport input** setting to ensure it supports SSH, or Telnet, or both, depending on the scenario. As a strategy for the exam, look for the command to confirm its settings when looking at any Telnet or SSH configuration. Also, be aware of the traditional defaults: Many older switches defaulted to **transport input all**, while older routers defaulted to **transport input none**, with more recent Cisco switches and routers now defaulting to the more-secure **transport input ssh**.

To complete this section about SSH, the following configuration checklist details the steps for one method to configure a Cisco switch to support SSH using local usernames. (SSH support in IOS can be configured in several ways; this checklist shows one simple way to configure it.) The process shown here ends with a comment to configure local username support on vty lines, as was discussed earlier in the section titled “Securing User Mode Access with Local Usernames and Passwords.”

Config Checklist

Step 1. Configure the switch to generate a matched public and private key pair to use for encryption:

- **a.** If not already configured, use the **hostname** *name* in global configuration mode to configure a hostname for this switch.

- **b.** If not already configured, use the **ip domain name** *name* in global configuration mode to configure a domain name for the switch, completing the switch's FQDN.
- **c.** Use the **crypto key generate rsa** command in global configuration mode (or the **crypto key generate rsa modulus** *modulus-value* command to avoid being prompted for the key modulus) to generate the keys. (Use at least a 768-bit key to support SSH version 2.)

Step 2. (Optional) Use the **ip ssh version 2** command in global configuration mode to override the default of supporting both versions 1 and 2, so that only SSHv2 connections are allowed.

Step 3. (Optional) If not already configured with the setting you want, configure the vty lines to accept SSH and whether to also allow Telnet:

- **a.** Use the **transport input ssh** command in vty line configuration mode to allow SSH only.
- **b.** Use the **transport input all** command or **transport input telnet ssh** command in vty line configuration mode to allow both SSH and Telnet.

Step 4. Use various commands in vty line configuration mode to configure local username login authentication as discussed earlier in this chapter.

Two key commands give some information about the status of SSH on the switch. First, the **show ip ssh** command lists status information about the SSH server itself. The **show ssh** command then lists information about each SSH client currently connected to the switch. Example 6-6 shows samples of each, with user wendell currently connected to the switch.

Example 6-6 *Displaying SSH Status*

```
SW1# show ip ssh
SSH Enabled - version 2.0
Authentication timeout: 120 secs; Authentication retries: 3

SW1# show ssh
Connection Version Mode Encryption    Hmac           State              Username
0          2.0     IN   aes126-cbc    hmac-sha1      Session started    wendell
0          2.0     OUT  aes126-cbc    hmac-sha1      Session started    wendell
%No SSHv1 server connections running.
```

Enabling and Securing the WebUI

The section "Accessing the CLI with the WebUI" in Chapter 4 shows examples of the WebUI user interface but does not discuss any configurations. Next, you will discover how to configure the most common settings to support this feature and to secure it using a **username** command—but with a new twist.

The HTTP server, the WebUI, has a long history as an integrated IOS feature in switches and routers. Over time, Cisco changed the user interface and some configuration commands. So, rather than focus on the trivial matters, focus on these steps that would be a common-sense configuration in a switch (or router) today:

- Use the **no ip http server** global command to disable the HTTP server (port 80) (traditionally enabled by default).
- Use the **ip http secure-server** global command to enable the HTTPS server (port 443, uses TLS) (traditionally enabled by default).
- Use the **ip http authentication local** global command to define the authentication method to use locally defined usernames (traditionally defaults to use the enable password).
- Use the **username** *name* **priority 15 password** *pass-value* global command to define one or more usernames with privilege level 15.

Talking through the items in the list, IOS has long used defaults that enable both the HTTP and HTTPS servers. With both enabled, users can connect from browsers by typing URLs that begin http:// (therefore not encrypting the traffic) or https:// (therefore encrypting the traffic). For better security, Cisco recommends disabling the HTTP server.

The HTTP server has long allowed three options to log in to a device from a web browser:

- Using the enable password
- Using a local username/password
- Using the AAA settings on the device

More recent IOS versions move away from the **enable** option. Also, using both a username and a password makes more sense from a security perspective.

To access all the features of the HTTP server (the WebUI), when using local usernames for authentication, you must configure a privilege level of 15 for the user. IOS internally defines user privilege levels, by default creating two levels, 0 and 15. IOS assigns user mode to level 0 and privileged mode (enable mode) to level 15. When CLI users move from user mode to enable mode, they improve their priority level from 0 to 15.

For the WebUI, if you log in using a username with the **privilege 15** option, you receive access to all WebUI features, including the ability to use the CLI configuration mode, install new software, erase the configuration, and reload the router. So, the **username** *name* **priority 15 password** *pass-value* global command creates a way to enter privileged mode immediately. (If you omit the **priority 15** option and log in to the WebUI with that username you can log in, but you cannot do advanced features, including using the CLI to configure or verify a feature.)

Enabling IPv4 for Remote Access

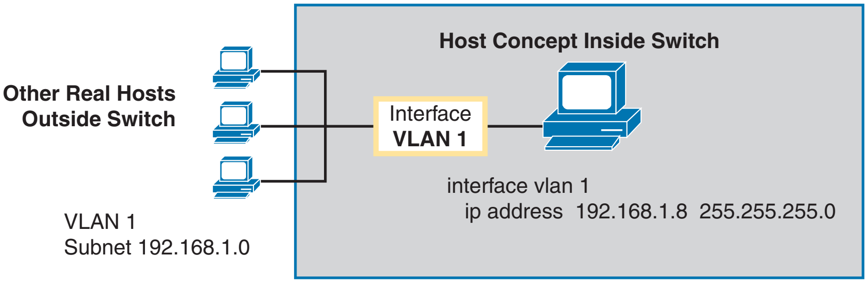
To allow Telnet, SSH, or WebUI access to the switch, and to allow other IP-based management protocols (for example, Simple Network Management Protocol, or SNMP) to function as intended, the switch needs an IP address, as well as a few other related settings. The IP address has nothing to do with how switches forward Ethernet frames; it simply exists to support overhead management traffic (control plane traffic).

This next topic begins by explaining the IPv4 settings needed on a switch, followed by the configuration. Note that although switches can be configured with IPv6 addresses with commands similar to those shown in this chapter, this chapter focuses solely on IPv4. All references to IP in this chapter imply IPv4.

Host and Switch IP Settings

A switch needs the same kind of IP settings as a PC with a single Ethernet interface. For perspective, a PC has a CPU, with the operating system running on the CPU. It has an Ethernet network interface card (NIC). The OS configuration includes an IP address associated with the NIC, either configured or learned dynamically with Dynamic Host Configuration Protocol (DHCP).

A switch uses the same ideas, except that the switch needs to use a virtual NIC inside the switch. Like a PC, a switch has a real CPU, running an OS (called IOS). The switch obviously has lots of Ethernet ports, but instead of assigning its management IP address to any of those ports, the switch then uses a NIC-like concept called a switch virtual interface (SVI), or more commonly, a **VLAN interface**, that acts like the switch's own NIC. Then the settings on the switch look something like a host, with the switch configuration assigning IP settings, like an IP address, to this VLAN interface, as shown in Figure 6-6.


Host Concept Inside Switch
Other Real Hosts Outside Switch
Interface VLAN 1
interface vlan 1
ip address 192.168.1.8 255.255.255.0
VLAN 1
Subnet 192.168.1.0

(Shaded Area Is Inside the Switch)

Figure 6-6 *Switch Virtual Interface (SVI) Concept Inside a Switch*

By using interface VLAN 1 for the IP configuration, the switch can then send and receive frames on any of the ports in VLAN 1. In a Cisco switch, by default, all ports are assigned to VLAN 1.

In most networks, switches configure many VLANs, so the network engineer has a choice of where to configure the IP address. That is, the management IP address does not have to be configured on the VLAN 1 interface (as configured with the **interface vlan 1** command seen in Figure 6-6).

A Layer 2 Cisco LAN switch needs only one IP address for management purposes. However, you can choose to use any VLAN to which the switch connects. The configuration then includes a VLAN interface for that VLAN number, with an appropriate IP address.

For example, Figure 6-7 shows a Layer 2 switch with some physical ports in two different VLANs (VLANs 1 and 2). The figure also shows the subnets used on those VLANs. The network engineer could choose to use either

- Interface VLAN 1, with an IP address in subnet 192.168.1.0
- Interface VLAN 2, with an IP address in subnet 192.168.2.0

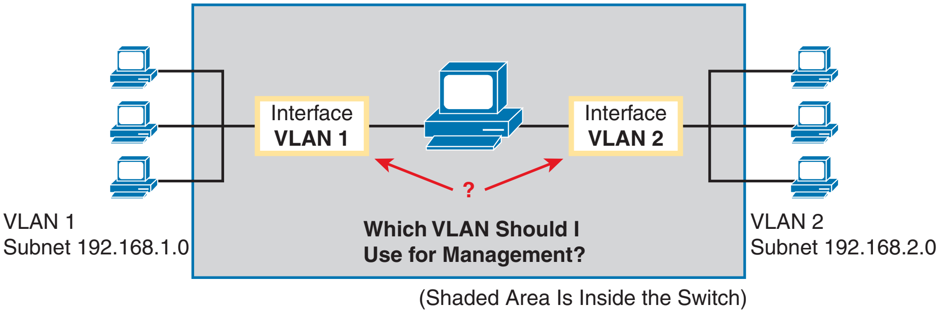


Figure 6-7 *Choosing One VLAN on Which to Configure a Switch IP Address*

Note that you should not try to use a VLAN interface for which there are no physical ports assigned to the same VLAN. If you do, the VLAN interface will not reach an up/up state, and the switch will not have the physical ability to communicate outside the switch.

NOTE Some Cisco switches can be configured to act as either a Layer 2 switch or a Layer 3 switch. When acting as a Layer 2 switch, a switch forwards Ethernet frames as discussed in depth in Chapter 5, "Analyzing Ethernet LAN Switching." Alternatively, a switch can also act as a *multilayer switch* or *Layer 3 switch*, which means the switch can do both Layer 2 switching and Layer 3 IP routing of IP packets, using the Layer 3 logic normally used by routers. This chapter assumes all switches are Layer 2 switches. Chapter 18, "IP Routing in the LAN," discusses Layer 3 switching in depth along with using multiple VLAN interfaces at the same time.

Configuring the IP address (and mask) on one VLAN interface allows the switch to send and receive IP packets with other hosts in a subnet that exists on that VLAN; however, the switch cannot communicate outside the local subnet without another configuration setting called the **default gateway**. The reason a switch needs a default gateway setting is the same reason that hosts need the same setting—because of how hosts think when sending IP packets. Specifically:

- To send IP packets to hosts in the same subnet, send them directly
- To send IP packets to hosts in a different subnet, send them to the local router; that is, the default gateway

Figure 6-8 shows the ideas. In this case, the switch (on the right) will use IP address 192.168.1.200 as configured on interface VLAN 1. However, to communicate with host A, on the far left of the figure, the switch must use Router R1 (the default gateway) to forward IP packets to host A. To make that work, the switch needs to configure a default gateway setting, pointing to Router R1's IP address (192.168.1.1 in this case). Note that the switch and router both use the same mask, 255.255.255.0, which puts the addresses in the same subnet.

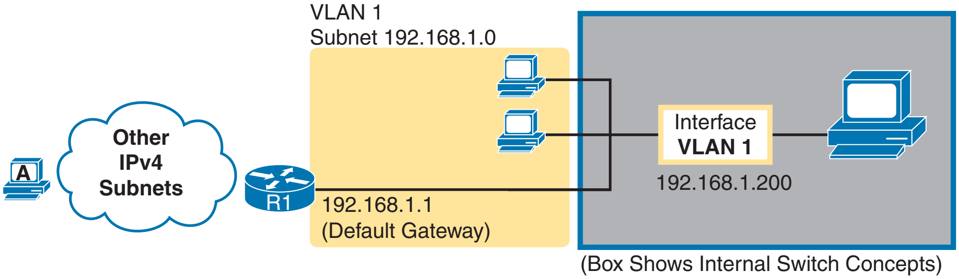


Figure 6-8 *The Need for a Default Gateway*

Configuring IPv4 on a Switch

A switch configures its IPv4 address and mask on this special NIC-like *VLAN interface*. The following steps list the commands used to configure IPv4 on a switch, assuming that the IP address is configured to be in VLAN 1, with Example 6-7 that follows showing an example configuration.

Config Checklist

Step 1. Use the **interface vlan 1** command in global configuration mode to enter interface VLAN 1 configuration mode.

Step 2. Use the **ip address** *ip-address mask* command in interface configuration mode to assign an IP address and mask.

Step 3. Use the **no shutdown** command in interface configuration mode to enable the VLAN 1 interface if it is not already enabled.

Step 4. Add the **ip default-gateway** *ip-address* command in global configuration mode to configure the default gateway.

Step 5. (Optional) Add the **ip name-server** *ip-address1 ip-address2* … command in global configuration mode to configure the switch to use the Domain Name System (**DNS**) to resolve names into their matching IP address.

Example 6-7 *Switch Static IP Address Configuration*

```
Emma# configure terminal
Emma(config)# interface vlan 1
Emma(config-if)# ip address 192.168.1.200 255.255.255.0
Emma(config-if)# no shutdown
00:25:07: %LINK-3-UPDOWN: Interface Vlan1, changed state to up
00:25:08: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed
 state to up
Emma(config-if)# exit
Emma(config)# ip default-gateway 192.168.1.1
```

On a side note, this example shows a particularly important and common command: the [**no**] **shutdown** command. To administratively enable an interface on a switch, use the **no shutdown** interface subcommand; to disable an interface, use the **shutdown** interface subcommand.

This command can be used on the physical Ethernet interfaces that the switch uses to switch Ethernet messages in addition to the VLAN interface shown here in this example.

Also, pause long enough to look at the messages that appear just below the **no shutdown** command in Example 6-7. Those messages are syslog messages generated by the switch stating that the switch did indeed enable the interface. Switches (and routers) generate syslog messages in response to a variety of events, and by default, those messages appear at the console. Chapter 13, "Device Management Protocols," in the *CCNA 200-301 Official Cert Guide, Volume 2*, Second Edition, discusses syslog messages in more detail.

Configuring a Switch to Learn Its IP Address with DHCP

The switch can also use Dynamic Host Configuration Protocol (DHCP) to dynamically learn its IPv4 settings. (Typically, engineers do not do so, instead statically configuring switch IP addresses, but this section covers the concept to be complete as compared to the exam topics.) Basically, all you have to do is tell the switch to use DHCP on the interface and enable the interface. Assuming that DHCP works in this network, the switch will learn all its settings. The following list details the steps, again assuming the use of interface VLAN 1, with Example 6-8 that follows showing an example:

Config Checklist

Step 1. Enter VLAN 1 configuration mode using the **interface vlan 1** global configuration command, and enable the interface using the **no shutdown** command as necessary.

Step 2. Assign an IP address and mask using the **ip address dhcp** interface subcommand.

Example 6-8 *Switch Dynamic IP Address Configuration with DHCP*

```
Emma# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Emma(config)# interface vlan 1
Emma(config-if)# ip address dhcp
Emma(config-if)# no shutdown
Emma(config-if)# ^Z
Emma#
00:38:20: %LINK-3-UPDOWN: Interface Vlan1, changed state to up
00:38:21: %LINEPROTO-5-UPDOWN: Line protocol on Interface Vlan1, changed state to up
```

Verifying IPv4 on a Switch

The switch IPv4 configuration can be checked in several places. First, you can always look at the current configuration using the **show running-config** command. Second, you can look at the IP address and mask information using the **show interfaces vlan** *x* command, which shows detailed status information about the VLAN interface in VLAN *x*. Finally, if using DHCP, use the **show dhcp lease** command to see the (temporarily) leased IP address and other parameters. (Note that the switch does not store the DHCP-learned IP configuration in the running-config file.) Example 6-9 shows sample output from these commands to match the configuration in Example 6-8.

Example 6-9 *Verifying DHCP-Learned Information on a Switch*

```
Emma# show dhcp lease
Temp IP addr: 192.168.1.101   for peer on Interface: Vlan1
Temp sub net mask: 255.255.255.0
   DHCP Lease server: 192.168.1.1, state: 3 Bound
   DHCP transaction id: 1966
   Lease: 86400 secs,  Renewal: 43200 secs,  Rebind: 75600 secs
Temp default-gateway addr: 192.168.1.1
   Next timer fires after: 11:59:45
   Retry count: 0   Client-ID: cisco-0019.e86a.6fc0-Vl1
   Hostname: Emma
Emma# show interfaces vlan 1
Vlan1 is up, line protocol is up
  Hardware is EtherSVI, address is 0019.e86a.6fc0 (bia 0019.e86a.6fc0)
  Internet address is 192.168.1.101/24
  MTU 1500 bytes, BW 1000000 Kbit, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
! lines omitted for brevity
Emma# show ip default-gateway
192.168.1.1
```

The output of the **show interfaces vlan 1** command lists two important details related to switch IP addressing. First, this **show** command lists the interface status of the VLAN 1 interface—in this case, "up and up." If the VLAN 1 interface is not up, the switch cannot use its IP address to send and receive management traffic. Notably, if you forget to issue the **no shutdown** command, the VLAN 1 interface remains in its default shutdown state and is listed as "administratively down" in the **show** command output.

Second, note that the output lists the interface's IP address on the third line. If you statically configure the IP address, as in Example 6-7, the IP address will always be listed; however, if you use DHCP and DHCP fails, the **show interfaces vlan** *x* command will not list an IP address here. When DHCP works, you can see the IP address with the **show interfaces vlan 1** command, but that output does not remind you whether the address is either statically configured or DHCP leased. So it does take a little extra effort to make sure you know whether the address is statically configured or DHCP-learned on the VLAN interface.

Miscellaneous Settings Useful in the Lab

This last short section of the chapter touches on a couple of commands that can help you be a little more productive when practicing in a lab.

History Buffer Commands

When you enter commands from the CLI, the switch saves the last several commands in the **history buffer**. Then, as mentioned in Chapter 4, "Using the Command-Line Interface," you can use the up-arrow key or press Ctrl+P to move back in the history buffer to retrieve a command you entered a few commands ago. This feature makes it easy and fast to use a set of commands repeatedly. Table 6-2 lists some of the key commands related to the history buffer.

Table 6-2 Commands Related to the History Buffer

Command	Description
show history	An EXEC command that lists the commands currently held in the history buffer.
terminal history size *x*	From EXEC mode, this command allows a single user to set, just for this one login session, the size of his or her history buffer.
history size *x*	A configuration command that, from console or vty line configuration mode, sets the default number of commands saved in the history buffer for the users of the console or vty lines, respectively.

The logging synchronous, exec-timeout, and no ip domain-lookup Commands

These next three configuration commands have little in common, other than the fact that they can be useful settings to reduce your frustration when using the console of a switch or router.

The console automatically receives copies of all unsolicited syslog messages on a switch. The idea is that if the switch needs to tell the network administrator some important and possibly urgent information, the administrator might be at the console and might notice the message.

Unfortunately, IOS (by default) displays these syslog messages on the console's screen at any time—including right in the middle of a command you are entering, or in the middle of the output of a **show** command. Having a bunch of text show up unexpectedly can be a bit annoying.

You could simply disable the feature that sends these messages to the console and then re-enable the feature later using the **no logging console** and **logging console** global configuration commands. For example, when working from the console, if you want to temporarily not be bothered by log messages, you can disable the display of these messages with the **no logging console** global configuration command, and then when finished, enable them again.

However, IOS supplies a reasonable compromise, telling the switch to display syslog messages only at more convenient times, such as at the end of output from a **show** command. To do so, just configure the **logging synchronous** console line subcommand, which basically tells IOS to synchronize the syslog message display with the messages requested using **show** commands.

Another way to improve the user experience at the console is to control timeouts of the login session from the console or when using Telnet or SSH. By default, the switch automatically disconnects console and vty (Telnet and SSH) users after 5 minutes of inactivity. The **exec-timeout** *minutes seconds* line subcommand enables you to set the length of that inactivity timer. In the lab (but not in production), you might want to use the special value of 0 minutes and 0 seconds, meaning "never time out."

Finally, IOS has an interesting combination of features that can make you wait for a minute or so when you mistype a command. First, IOS tries to use DNS **name resolution** on IP hostnames—a generally useful feature. If you mistype a command, however, IOS thinks you want to telnet to a host by that name. With all default settings in the switch, the switch tries to resolve the hostname, cannot find a DNS server, and takes about a minute to time out and give you control of the CLI again.

To avoid this problem, configure the **no ip domain-lookup** global configuration command, which disables IOS's attempt to resolve the hostname into an IP address.

Example 6-10 collects all these commands into a single example, as a template for some good settings to add in a lab switch to make you more productive.

Example 6-10 *Commands Often Used in the Lab to Increase Productivity*

```
no ip domain-lookup
!
line console 0
 exec-timeout 0 0
 logging synchronous
 history size 20
!
line vty 0 15
 exec-timeout 0 0
 logging synchronous
 history size 20
```

Chapter Review

One key to doing well on the exams is to perform repetitive spaced review sessions. Review this chapter's material using either the tools in the book or interactive tools for the same material found on the book's companion website. Refer to the "Your Study Plan" element section titled "Step 2: Build Your Study Habits Around the Chapter" for more details. Table 6-3 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 6-3 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP
Review config checklists		Book, website
Do labs		Sim Lite, blog
Review command tables		Book
Watch video		Website

Review All the Key Topics

Key Topic

Table 6-4 Key Topics for Chapter 6

Key Topic Element	Description	Page Number
Example 6-2	Example of configuring password login security (no usernames)	138
Figure 6-5	SSH configuration commands with related username login security	142

Key Terms You Should Know

AAA, AAA server, default gateway, DNS, enable mode, history buffer, local username, log message, name resolution, Secure Shell (SSH), Telnet, VLAN interface

Do Labs

The Sim Lite software is a version of Pearson's full simulator learning product with a subset of the labs, included with this book for free. The subset of labs mostly relates to this chapter. Take the time to try some of the labs. As always, also check the author's blog site pages for configuration exercises (Config Labs) at https://www.certskills.com.

Command References

Tables 6-5, 6-6, 6-7, and 6-8 list configuration and verification commands used in this chapter. As an easy review exercise, cover the left column in a table, read the right column, and try to recall the command without looking. Then repeat the exercise, covering the right column, and try to recall what the command does.

Table 6-5 Login Security Commands

Command	Mode/Purpose/Description
line console 0	Changes the context to console configuration mode.
line vty *1st-vty last-vty*	Changes the context to vty configuration mode for the range of vty lines listed in the command.
login	Console and vty configuration mode. Tells IOS to prompt for a password.
password *pass-value*	Console and vty configuration mode. Lists the password required if the **login** command (with no other parameters) is configured.
login local	Console and vty configuration mode. Tells IOS to prompt for a username and password, to be checked against locally configured **username** global configuration commands on this switch or router.
username *name* **secret** *pass-value*	Global command. Defines one of possibly multiple usernames and associated passwords, used for user authentication. Used when the **login local** line configuration command has been used.

Command	Mode/Purpose/Description
crypto key generate rsa [modulus *512..2048*]	Global command. Creates and stores (in a hidden location in flash memory) the keys required by SSH.
transport input {**telnet** | **ssh** | **all** | **none**}	vty line configuration mode. Defines whether Telnet/SSH access is allowed into this switch. Both values can be configured on one command to allow both Telnet and SSH access (the default).
ip domain name *fqdn*	Global command. Defines the fully-qualified domain name (*fqdn*) for the DNS domain in which the switch or router resides.
hostname *name*	Global command. Sets the *name* that the device uses for itself, which is also used at the initial text in the command prompt.
ip ssh version 2	Global command. Sets the SSH server to use only version 2, rather than the default of supporting both versions 1 and 2.

Table 6-6 Switch IPv4 Configuration

Command	Mode/Purpose/Description
interface vlan *number*	Changes the context to VLAN interface mode. For VLAN 1, allows the configuration of the switch's IP address.
ip address *ip-address subnet-mask*	VLAN interface mode. Statically configures the switch's IP address and mask.
ip address dhcp	VLAN interface mode. Configures the switch as a DHCP client to discover its IPv4 address, mask, and default gateway.
ip default-gateway *address*	Global command. Configures the switch's default gateway IPv4 address. Not required if the switch uses DHCP.
ip name-server *server-ip-1 server-ip-2 …*	Global command. Configures the IPv4 addresses of DNS servers, so any commands when logged in to the switch will use the DNS for name resolution.

Table 6-7 Other Switch Configuration

Command	Mode/Purpose/Description
hostname *name*	Global command. Sets this switch's hostname, which is also used as the first part of the switch's command prompt.
enable secret *pass-value*	Global command. Sets this switch's password that is required for any user to reach enable mode.
history size *length*	Line config mode. Defines the number of commands held in the history buffer, for later recall, for users of those lines.
logging synchronous	Console or vty mode. Tells IOS to send log messages to the user at natural break points between commands rather than in the middle of a line of output.
[no] logging console	Global command. Disables or enables the display of log messages to the console.

Command	Mode/Purpose/Description
exec-timeout *minutes* [*seconds*]	Console or vty mode. Sets the inactivity timeout so that after the defined period of no action, IOS closes the current user login session.
no ip domain-lookup	Global command. Disables the use of the DNS client on the switch.

Table 6-8 Chapter 6 EXEC Command Reference

Command	Purpose
show running-config	Lists the currently used configuration.
show running-config | begin line vty	Pipes (sends) the command output to the **begin** command, which only lists output beginning with the first line that contains the text "line vty."
show dhcp lease	Lists any information the switch acquires as a DHCP client. This includes IP address, subnet mask, and default gateway information.
show crypto key mypubkey rsa	Lists the public and shared key created for use with SSH using the **crypto key generate rsa** global configuration command.
show ip ssh	Lists status information for the SSH server, including the SSH version.
show ssh	Lists status information for current SSH connections into and out of the local switch.
show interfaces vlan *number*	Lists the interface status, the switch's IPv4 address and mask, and much more.
show ip default-gateway	Lists the switch's setting for its IPv4 default gateway.
terminal history size *x*	Changes the length of the history buffer for the current user only, only for the current login to the switch.
show history	Lists the commands in the current history buffer.

This page intentionally left blank

CHAPTER 7

Configuring and Verifying Switch Interfaces

This chapter covers the following exam topics:

- **1.0 Network Fundamentals**
  - **1.1 Explain the role and function of network components**
    - **1.1.b Layer 2 and Layer 3 switches**
  - **1.3 Compare physical interface and cabling types**
    - **1.3.b Connections (Ethernet shared media and point-to-point)**
  - **1.4 Identify interface and cable issues (collisions, errors, mismatch duplex, and/or speed)**

The chapters in Part II of this book move back and forth between switch administration and core switch functions. In Chapter 4, “Using the Command-Line Interface,” you read about the fundamentals of the command-line interface (CLI) and how to use commands that configure and verify switch features. In Chapter 5, “Analyzing Ethernet LAN Switching,” you learned about the primary purpose of a switch—forwarding Ethernet frames—and how to see that process in action by looking at the switch MAC address table. Chapter 6, “Configuring Basic Switch Management,” then moved back to more administrative tasks, where you learned a few management plane features, like how to configure the switch to support Telnet and Secure Shell (SSH) by configuring IP address and login security.

This chapter focuses on more core switch features, specifically how to configure switch interfaces so that they work. The first section shows how to configure switch interfaces to use the correct speed and duplex, primarily by using IEEE autonegotiation. The second section examines some administrative settings on switches, including how to disable and re-enable an interface. The final section then focuses on how to use **show** commands on a switch to verify switch interface status and interpret the output to find some of the more common issues with switch interfaces.

“Do I Know This Already?” Quiz

Take the quiz (either here or use the PTP software) if you want to use the score to help you decide how much time to spend on this chapter. The letter answers are listed at the bottom of the page following the quiz. Appendix C, found both at the end of the book and on the companion website, includes answers and explanations. You can also find both answers and explanations in the PTP testing software.

Table 7-1 "Do I Know This Already?" Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Configuring Switch Interface Speed and Duplex	1–3
Managing Switch Interface Configuration	4–5
Analyzing Switch Interface Status and Statistics	6–8

1. Switch SW1 connects its G1/0/1 port to PC1. Both devices use IEEE autonegotiation and have 10/100/1000 ports. Which answer describes how the switch chooses its G1/0/1 speed and duplex settings? (Choose two answers.)
   - **a.** Speed by comparing capabilities per received autonegotiation messages
   - **b.** Speed by analyzing the electrical signal of incoming Ethernet frames from PC1
   - **c.** Duplex by comparing capabilities per received autonegotiation messages
   - **d.** Duplex by analyzing the electrical signal of incoming Ethernet frames from PC1
   - **e.** Duplex by choosing a default based on the chosen speed
2. In which of the following modes of the CLI could you configure the duplex setting for interface Fast Ethernet 0/5?
   - **a.** User mode
   - **b.** Enable mode
   - **c.** Global configuration mode
   - **d.** VLAN mode
   - **e.** Interface configuration mode
3. Switch SW1 connects its G1/0/1 port to PC1. Both devices are 10/100/1000 ports. While the switch port uses IEEE autonegotiation, PC1 has disabled it. Which answer describes how the switch chooses its G1/0/1 speed and duplex settings?
   - **a.** Speed by comparing capabilities per received autonegotiation messages
   - **b.** Speed by analyzing the electrical signal of incoming Ethernet frames from PC1
   - **c.** Duplex by comparing capabilities per received autonegotiation messages
   - **d.** Duplex by analyzing the electrical signal of incoming Ethernet frames from PC1
   - **e.** Duplex by choosing a default based on the chosen speed
4. Switch interface G1/0/5 has been cabled correctly in anticipation of some weekend work. However, the engineer needs to prevent the interface from being used until she enables it remotely during a change window this weekend. Which action helps achieve that goal?
   - **a.** Unplug the cable.
   - **b.** Issue the **shutdown g1/0/5** global configuration command.
   - **c.** Issue the **shutdown** interface subcommand under interface **g1/0/5**.
   - **d.** Issue the **disable g1/0/5** global configuration command.
   - **e.** Issue the **enable** interface subcommand under interface **g1/0/5**.

5. An engineer configures Cisco switch SW1 with the commands **interface range G1/0/10-20** and then **description connected to endpoint device**. The engineer exits configuration mode and issues a **show running-config** command. Which answers best describe the related output? (Choose two answers.)

- **a.** The output includes the **interface range** command.
- **b.** The output does not include the **interface range** command.
- **c.** The output lists one **description connected to endpoint device** interface subcommand.
- **d.** The output lists 11 **description connected to endpoint device** interface subcommands.

6. The output from the switch command **show interfaces status** shows interface Fa0/1 in a "disabled" state. Which of the following is true about interface Fa0/1? (Choose three answers.)

- **a.** The interface is configured with the **shutdown** command.
- **b.** The **show interfaces fa0/1** command will list the interface with two status codes of administratively down and line protocol down.
- **c.** The **show interfaces fa0/1** command will list the interface with two status codes up and down.
- **d.** The interface cannot currently be used to forward frames.
- **e.** The interface can currently be used to forward frames.

7. Switch SW1 Gigabit 1/0/1 connects to switch SW2's Gigabit 1/0/2 interface, both 10/100/1000 ports. The switch SW2 configuration includes the **speed** and **duplex** commands, the combination of which happens to disable autonegotiation on that port. Which combination of settings in SW2's **speed** and **duplex** commands results in a duplex mismatch between SW1 and SW2? (Choose two answers.)

- **a.** **speed 100** and **duplex full**
- **b.** **speed 100** and **duplex half**
- **c.** **speed 10** and **duplex full**
- **d.** **speed 10** and **duplex half**

8. Switch SW1 connects via a cable to switch SW2's G1/0/1 port. Which of the following conditions is the most likely to cause SW1's late collision counter to continue to increment?

- **a.** SW2's G1/0/1 has been configured with a **shutdown** interface subcommand.
- **b.** The two switches have been configured with different values on the **speed** interface subcommand.
- **c.** A duplex mismatch exists with SW1 set to full duplex.
- **d.** A duplex mismatch exists with SW1 set to half duplex.

Foundation Topics

Configuring Switch Interface Speed and Duplex

When physically creating an Ethernet LAN, you must consider the cabling and connectors that match the dozens of physical layer Ethernet standards that help you meet the physical requirements for the LAN. Once chosen and installed, any interfaces connected to fiber cabling require no additional configuration; however, UTP cabling can have different pinouts, and UTP supports different speeds and duplex settings. So once installed, interfaces that use UTP cabling may need additional configuration.

This first major section of the chapter examines interface speed and duplex settings, along with the IEEE autonegotiation process. It also discusses auto-MDIX, a feature that deals with pinout issues. Finally, this entire chapter continues the goal of helping you learn more about CLI navigation and conventions.

IEEE Autonegotiation Concepts

Ethernet NICs and switch ports often support multiple standards and therefore support multiple speeds. For instance, you will see designations like these:

10/100: A port that supports 10- and 100-Mbps Ethernet

10/100/1000: A port that supports 10-, 100-, and 1000-Mbps Ethernet

Using hardware that supports multiple standards and speeds allows for much easier growth and migration over time—mainly because the devices can automatically sense the fastest speed using the IEEE autonegotiation feature. For instance, many switches today have many 10/100/1000 ports. Many newer end-user devices also have a 10/100/1000 Ethernet NIC, so the PC and switch can autonegotiate to 1000 Mbps (1 Gbps). Older devices, and some specialized devices that might not need faster speeds, might support only 10 Mbps or 100 Mbps. **Autonegotiation** gives the devices on each link the means to agree to use the best speed without manually configuring the speed on each switch port.

IEEE autonegotiation defines a process by which both devices on each link tell the neighboring device their capabilities. Once both endpoints learn about the other, they use the standard with the fastest speed. Autonegotiation also defines **full duplex** as the preferred option over **half duplex**, assuming both devices support that option.

Autonegotiation Under Working Conditions

With IEEE autonegotiation, a device declares its capabilities by sending a series of Fast Link Pulses (FLPs). The data in the FLP messages include bits that identify the Ethernet standards supported by the device and the duplex ability supported, in effect declaring the speeds and duplex settings supported by the device.

The FLPs work even before the endpoints choose a physical layer standard. FLPs use out-of-band electrical signaling, independent of the various physical layer standards for Ethernet frame transmission. Any device that supports autonegotiation supports using these out-of-band FLP messages. The FLPs solve the problem of how the devices can send information to each other even before the link is up and working for normal data transmission.

Figure 7-1 shows three examples of autonegotiation working as intended. First, the company installs cabling that supports 10BASE-T, 100BASE-T, and 1000BASE-T, that is, cabling with four wire pairs of appropriate quality. In the three examples, the PC and switch both use autonegotiation. The cables work with correct straight-through pinouts. The switch, in this case, has all 10/100/1000 ports, while the PC NICs support different options (10 only, 10/100, and 10/100/1000), as per the figure.

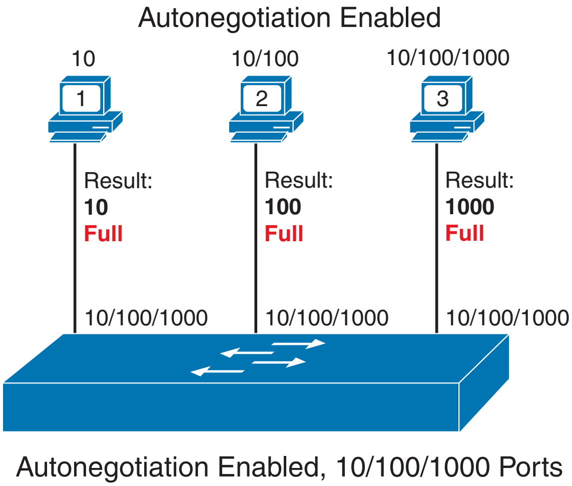


Figure 7-1 *IEEE Autonegotiation Results with Both Nodes Working Correctly*

The following list breaks down the logic, one PC at a time:

- **PC1:** PC1 sends autonegotiation FLPs that declare support for 10 Mbps only (not 100 or 1000 Mbps) and support for both full and half duplex. The switch announces support for 10, 100, and 1000 Mbps and both duplex settings. As a result, both the PC and the switch choose the fastest speed (10 Mbps) and the best duplex (full) that both support.
- **PC2:** PC2 declares support for 10 and 100 Mbps and both full and half duplex. The switch again claims support for 10, 100, and 1000 Mbps and both full and half duplex. Both devices use the best common speed and duplex (100 Mbps and full duplex).
- **PC3:** PC3 uses a 10/100/1000 NIC, supporting all three speeds, so both the NIC and switch port choose 1000 Mbps and full duplex.

Summarizing, the following list details autonegotiation rules when both endpoints use it:

Key Topic

- Both endpoints send messages, out-of-band compared to any specific data transmission standard, using Fast Link Pulses (FLPs).

Answers to the “Do I Know This Already?” quiz:

1 A, C **2** E **3** B, E **4** C **5** B, D **6** A, B, D **7** A, C **8** D

- The messages declare all supported speed and duplex combinations.
- After hearing from the link partner, each device chooses the fastest speed supported by both devices and the best duplex (full being better than half duplex).

Autonegotiation Results When Only One Node Uses Autonegotiation

With both devices using autonegotiation, the result is obvious. Both devices use the fastest speed and best duplex supported by both devices.

Cisco repeatedly recommends using autonegotiation on both ends of all Ethernet links that support it—but if you must disable it, make sure to configure both speed and duplex on both ends of the link. For instance, some installations prefer to predefine the speed and duplex on links between two switches. To do so, use commands such as **speed 1000** (meaning 1000 Mbps, or 1 Gbps) and **duplex full**. If configured with these same values on both ends of the link, the link will work due to matching settings of 1000BASE-T with full duplex.

However, many devices have the capability to disable autonegotiation, which means an engineer can make the poor choice to disable autonegotiation on one end of the link but not the other. In real networks, do not do that, because it can cause problems like a *duplex mismatch* and *late collisions*.

To understand what happens, first consider the device with autonegotiation disabled. It must use some physical layer standard (typically due to a static configuration setting), so it begins sending Ethernet frames that conform to that standard—but it does not send FLPs. The other device (the one that uses autonegotiation) sends FLPs, but receives none. Once the device that is attempting autonegotiation realizes it is not receiving FLPs, it can examine the incoming electrical signal of those Ethernet frames and notice the differences between the signals used for 10BASE-T, 100BASE-T, and so on in the incoming signals. Knowing that, it uses the same standard, solving the question of what speed to use.

The choice of duplex requires using a default. If the speed is 10 or 100 Mbps, the device attempting autonegotiation uses half duplex. Otherwise, it chooses full duplex.

The IEEE refers to the logic used by autonegotiation when the other device has disabled autonegotiation as **parallel detection**, summarized as follows:

Key Topic

- **Speed:** Detect the neighboring device's physical layer standard by analyzing the neighbor's incoming frames. Use that speed.
- **Duplex:** Make a default choice based on speed—half duplex if the speed is 10 or 100 Mbps, and full duplex if faster.

NOTE Ethernet interfaces using speeds more than 1 Gbps always use full duplex.

Figure 7-2 shows three examples of autonegotiation parallel detection logic. In each case, the PC configuration has disabled autonegotiation while the switch (with all 10/100/1000 ports) continues to use autonegotiation. The top of the figure shows the configured settings on each PC NIC, with the choices made by the switch listed next to each switch port.

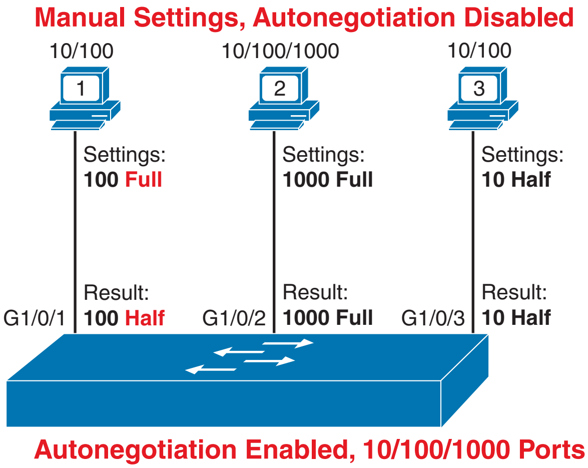


Figure 7-2 *IEEE Autonegotiation Results Using Parallel Detection Logic*

Reviewing each link, left to right:

- **PC1:** The PC uses 100 Mbps and full duplex settings. The switch receives no autonegotiation FLP messages on port G1/0/1, instead sensing that PC1 is sending frames at 100 Mbps. Then the switch chooses to use half duplex per the defaults (half duplex if the speed is 10 or 100 Mbps).
- **PC2:** The switch uses the same steps and logic as for the link to PC1. Switch port G1/0/2 senses the use of 1000BASE-T signaling, with a speed of 1 Gbps, and chooses a full duplex per the speed-based defaults.
- **PC3:** The PC uses the worst settings possible, with the slower speed (10 Mbps) and the worse duplex setting (half). The switch port receives no FLP messages, so it senses the use of 10BASE-T per the incoming frames, uses 10 Mbps, and chooses half duplex per the speed-based defaults.

Take a closer look at the PC1 example: It shows a poor result called a **duplex mismatch**. The two nodes (PC1 and SW1's port G1/0/1) both use the same 100 Mbps so that they can send data. However, PC1, using full duplex, does not attempt to use carrier sense multiple access with collision detection (CSMA/CD) logic and sends frames at any time. Switch port G1/0/1, using half duplex, does use CSMA/CD. As a result, switch port G1/0/1 will believe collisions occur on the link if, when sending a frame, PC1 also sends a frame. When that happens, the switch port will stop transmitting, back off, resend frames, and so on. As a result, the link is up, but it performs poorly. The upcoming section titled "The Duplex Mismatch Issue" will explore this problem with a focus on how to recognize the symptoms of a duplex mismatch.

NOTE To emphasize, in real networks, use autonegotiation. If you have specific reasons not to use it, ensure you configure the devices on both ends of the link and use the same settings.

Autonegotiation and LAN Hubs

LAN hubs also impact how autonegotiation works. Hubs do not participate in autonegotiation, they do not generate FLP messages, and they do not forward the autonegotiation FLP messages sent by connected devices. As a result, devices connected to a hub receive no FLP messages and use only IEEE autonegotiation parallel detection rules. That can work, but it often results in the devices using 10 Mbps and half duplex.

Figure 7-3 shows an example of a small Ethernet LAN that uses an old 10BASE-T hub. The devices on the right (PC3 and PC4) sense the speed as 10 Mbps per the incoming signal, and then they choose to use the default duplex when using 10 Mbps of half duplex. Using 10 Mbps and half duplex works well in this case: The PCs on the right need to use half duplex because the hub requires any attached devices to use half duplex to avoid collisions.

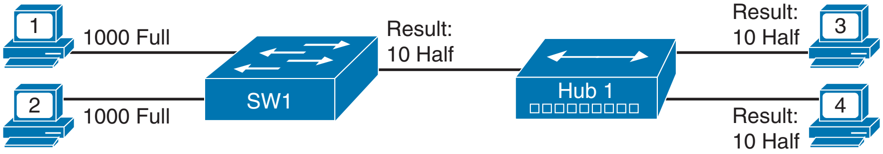


Figure 7-3 *IEEE Autonegotiation with a LAN Hub*

Configuring Autonegotiation, Speed, and Duplex

For an Ethernet link to work correctly, the link needs a working cable, and the endpoints need to use the same physical layer standard and duplex setting. With Cisco switches, the default switch setting to use autonegotiation should make the switch use the right speed and duplex so that the link works. The following pages show how to verify the autonegotiation process to see what a switch has chosen to use on an interface, along with how to manually set the speed and duplex with some other related commands.

Using Autonegotiation on Cisco Switches

Figure 7-4 shows a small network used in the next few examples. The figure shows a working link with an installed cable, a powered-on device (PC1), and both devices using IEEE autonegotiation. The other two links do not work because, in one case, no cable has been connected, and in the other, the cable is installed, but the device is powered off.

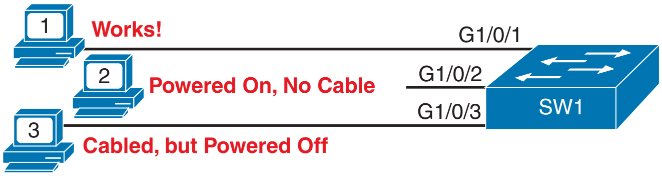


Figure 7-4 *Network Topology to Match Examples 7-1 Through 7-4*

The switch (SW1) uses the default autonegotiation settings in the first few examples. To configure those settings overtly, you would configure the interface subcommands **speed auto** and **duplex auto**. However, because the **show running-config** and **show startup-config** commands generally do not show default configuration commands, the absence of the **speed** and **duplex** commands in Example 7-1 confirms the interfaces use autonegotiation.

Key Topic

Example 7-1 *Confirming All Default Settings on the Switch Interfaces in Figure 7-1*

```
SW1# show running-config
! Lines omitted for brevity
interface GigabitEthernet1/0/1
!
interface GigabitEthernet1/0/2
!
interface GigabitEthernet1/0/3
!
! Lines omitted for brevity
SW1#
```

Example 7-2 shows how you can overcome your doubts about whether the absence of the **speed auto** and **duplex auto** interface subcommands confirms those settings. The example shows the commands configured on an interface. Even after configuring the commands, the **show running-config** command does not display them, confirming them as the default settings. (You can use that process for any configuration command, by the way.)

Example 7-2 *Confirming Autonegotiation Is the Default Setting*

```
SW1# configure terminal
SW1(config)# interface gigabitEthernet 1/0/1
SW1(config-if)# speed auto
SW1(config-if)# duplex auto
SW1(config-if)# ^Z
SW1# show running-config interface gigabitEthernet 1/0/1
Building configuration...

Current configuration : 38 bytes
!
interface GigabitEthernet1/0/1
end
SW1#
```

Also, as a quick way to check interface configuration, note the **show running-config interface GigabitEthernet1/0/1** command at the end of the example. It is a supported command that shows only the configuration for the listed interface.

The better way to confirm the operation of autonegotiation on a switch interface relies on the **show interfaces status** command. Example 7-3 shows the output based on the state from Figure 7-4. In particular, note the Duplex and Speed columns of the output, with the following list identifying the meaning:

a-full: Full duplex, with the a- meaning the switch learned the value using autonegotiation.

a-1000: 1000 Mbps (1 Gbps), with the a- meaning the switch learned the setting using autonegotiation.

auto: The interface will use autonegotiation when the link physically works.

Key Topic

Example 7-3 *Interpreting Autonegotiation Clues Before/After Completion*

```
SW1# show interfaces status
Port         Name            Status       Vlan       Duplex  Speed Type
Gi1/0/1                                   1
Gi1/0/2                                   1
Gi1/0/3                                   1
Gi1/0/4                      notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/5                      notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/6                      notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/7                      notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/8                      notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/9                      notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/10                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/11                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/12                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/13                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/14                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/15                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/16                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/17                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/18                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/19                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/20                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/21                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/22                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/23                     notconnect   1          auto     auto 10/100/1000BaseTX
Gi1/0/24                     notconnect   1          auto     auto 10/100/1000BaseTX
Te1/1/1                      connected    1          full      10G SFP-10GBase-SR
Te1/1/2                      notconnect   1          auto     auto unknown
Te1/1/3                      notconnect   1          auto     auto unknown
Te1/1/4                      notconnect   1          auto     auto unknown
```

The first few output lines in Example 7-3 confirm that port G1/0/1 works with autonegotiation. Given the highlighted values in the duplex and speed columns, you can see that port G1/0/1 uses 1000BASE-T and full duplex, as learned by autonegotiation (per the a- prefix).

The output lines for ports G1/0/2 and G1/0/3 show the normal state for a nonworking port. The notconnect state per the Status column means that the link is not functional—in PC2's case because no cable exists and in PC3's case because PC3 is powered off. Autonegotiation can work only if the physical link works.

Beyond the **show interfaces status**, the **show interfaces** command also gives some autonegotiation data. Example 7-4 shows the output for working interface G1/0/1 from Figure 7-4. Note the highlighted text about seven lines into the example, which shows the speed and duplex used on the link; however, note that this command on working interface G1/0/1 lists no information on whether it used autonegotiation.

Key Topic

Example 7-4 show interfaces *Command Autonegotiation Clues*

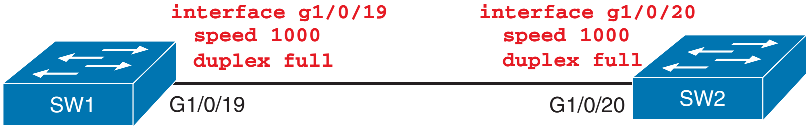
```
SW1# show interfaces gigabitEthernet 1/0/1
GigabitEthernet1/0/1 is up, line protocol is up (connected)
  Hardware is Gigabit Ethernet, address is 4488.165a.f201 (bia 4488.165a.f201)
  MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 1000Mb/s, media type is 10/100/1000BaseTX
  input flow-control is on, output flow-control is unsupported
  ARP type: ARPA, ARP Timeout 04:00:00
  Last input 00:00:01, output 00:00:00, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/2000/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 0 bits/sec, 0 packets/sec
  5 minute output rate 0 bits/sec, 0 packets/sec
     14228 packets input, 1870879 bytes, 0 no buffer
     Received 14223 broadcasts (14222 multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog, 14222 multicast, 0 pause input
     0 input packets with dribble condition detected
     56865 packets output, 7901974 bytes, 0 underruns
     Output 17109 broadcasts (0 multicasts)
     0 output errors, 0 collisions, 2 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 pause output
     0 output buffer failures, 0 output buffers swapped out

SW1# show interfaces g1/0/2
GigabitEthernet1/0/2 is down, line protocol is down (notconnect)
  Hardware is Gigabit Ethernet, address is 4488.165a.f202 (bia 4488.165a.f202)
  MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Auto-duplex, Auto-speed, media type is 10/100/1000BaseTX
! Lines omitted for brevity
```

However, the **show interfaces** command does give some insight into autonegotiation for a nonworking interface such as G1/0/2, as seen at the end of Example 7-4. On an interface with default settings **speed auto** and **duplex auto**, the command output implies those settings with the auto-duplex and auto-speed text. However, the output shows those specific settings if configured to a specific speed or duplex.

Setting Speed and Duplex Manually

The switch **speed** and **duplex** interface subcommands can set an interface's specific speed and duplex. Risking being repetitive: In production networks, use and trust autonegotiation. However, for exam preparation, to cover all the combinations, if you want to configure the settings instead, Cisco recommends that you configure both devices on the ends of the link (to the same values, of course).

Figure 7-5 and Example 7-5 show an example of manually configuring the speed and duplex on a link between two switches. As per the recommendation, the engineer configures both devices with the same settings, and the link works.



```
interface g1/0/19
 speed 1000
 duplex full
```

```
interface g1/0/20
 speed 1000
 duplex full
```

Figure 7-5 *Configuring Speed and Duplex on a Switch-to-Switch Link*

Example 7-5 *Configuring Speed and Duplex on Both Ends of a Link*

```
SW1# show running-config interface g1/0/19
Building configuration...

Current configuration : 63 bytes
!
interface GigabitEthernet1/0/19
 speed 1000
 duplex full
end
```

```
! Now, on switch SW2
SW2# show running-config interface g1/0/20
Building configuration...

Current configuration : 64 bytes
!
interface GigabitEthernet1/0/20
 speed 1000
 duplex full
end
SW1#
```

In the scenario shown in Example 7-5, the two switches set both the speed and duplex, so they do not need to use autonegotiation. In this case, they just begin using the 1000BASE-T standard with full duplex, and the link works.

Example 7-6 shows a hidden gem in the output of the **show interfaces status** command that tells you the switch did not use autonegotiation in this case. First, note that switch SW1's

G1/0/19 interface reaches a connected state, so the link works. The Duplex and Speed columns list full and 1000 without the "a-" prefix. The absence of the "a-" means that the interface did not use autonegotiation to choose the setting, instead using the configuration. (See Example 7-3 for the earlier example showing values of a-full and a-1000.)

Key Topic

Example 7-6 **show interfaces status** *Without Using Autonegotiation*

```
SW1# show interfaces g1/0/20 status

Port          Name              Status       Vlan       Duplex  Speed Type
Gi1/0/19                        connected    1            full   1000
10/100/1000BaseTX

SW1# show interfaces g1/0/19
GigabitEthernet1/0/19 is up, line protocol is up (connected)
  Hardware is Gigabit Ethernet, address is 4488.165a.f213 (bia 4488.165a.f213)
  MTU 1500 bytes, BW 1000000 Kbit/sec, DLY 10 usec,
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation ARPA, loopback not set
  Keepalive set (10 sec)
  Full-duplex, 1000Mb/s, media type is 10/100/1000BaseTX
! Lines omitted for brevity
```

NOTE On some Cisco Catalyst switch ports, configuring both speed and duplex disables autonegotiation on that port. On others, it does not. In real networks, should you ever need to configure both the speed and duplex to specific values, take extra care to confirm that the speed and duplex match on both ends of the link.

Using Auto-MDIX on Cisco Switches

For switch interfaces to work, meaning they reach a connected state, the engineer should install a correct cable between the switch port and some other connected device. For UTP cabling, the cables often terminate with RJ-45 connectors. The cables must also conform to a straight-through or crossover cable pinout, as discussed in the section, "Building Physical Ethernet LANs with UTP," in Chapter 2, "Fundamentals of Ethernet LANs."

Chapter 2 also introduced a related concept called **auto-MDIX**, or automatic medium-dependent interface crossover. Auto-MDIX, when enabled, gives an Ethernet interface the ability to sense when the attached cable uses the wrong cable pinout and to overcome the problem. For instance, a link between two switches should use a crossover cable pinout. If the cable has a straight-through pinout, the auto-MDIX feature can sense the issue and swap pairs in the interface electronics, achieving the same effect as a crossover cable.

Figure 7-6 and Examples 7-7 and 7-8 demonstrate auto-MDIX. The figure shows a case with two switches connected with a straight-through cable. Cisco Catalyst switches use auto-MDIX by default, with a default interface subcommand of **mdix auto**. As with the **speed** and **duplex** commands earlier in this chapter, the default interface subcommand **mdix auto** command does not appear in the configuration. Example 7-7 confirms the absence of the **mdix auto** subcommand but confirms that the link reaches a connected state.

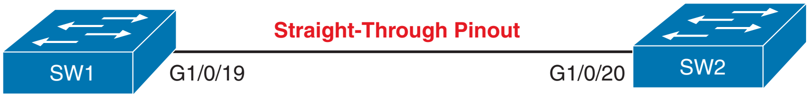


Figure 7-6 *Switch-Switch Link That Needs Crossover but Uses Straight-Through Cable*

Example 7-7 *Switch-Switch Link Works with All Default Settings*

```
SW1# running-config interface GigabitEthernet 1/0/19
Building configuration...

Current configuration : 39 bytes
!
interface GigabitEthernet1/0/19
end

SW1# show interfaces 1/0/19 status

Port         Name               Status       Vlan       Duplex  Speed Type
Gi1/0/19                        connected    1          a-full  a-1000 10/100/1000BaseTX
```

Auto-MDIX works if either one or both endpoints on the link enable auto-MDIX. In Example 7-7, both switches default to auto-MDIX, but only one needs to swap the wire pairs. If you disable auto-MDIX on just one side, the other side swaps the pairs. To prevent auto-MDIX from doing its valuable work, disable it on both ends of the link using the **no mdix auto** interface subcommand.

Example 7-8 shows just that example. Before the example, the engineer configured switch SW2 with the **no mdix auto** interface subcommand. The example shows the process to disable it on switch SW1. As a result, neither switch uses auto-MDIX, and the interface fails to a notconnect state.

Example 7-8 *Switch-Switch Link Fails with Auto-MDIX Disabled on Both Ends*

```
SW1# running-config interface GigabitEthernet 1/0/19
SW1# config t
Enter configuration commands, one per line.  End with CNTL/Z.
SW1(config)# int g1/0/19
SW1(config-if)# no mdix auto
SW1(config-if)#
*Oct  5 12:50:22.177: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/19, changed state to down
*Oct  5 12:50:23.175: %LINK-3-UPDOWN: Interface GigabitEthernet1/0/19, changed state
to down
SW1(config-if)# ^Z
SW1#
SW1# show interfaces g1/0/19 status

Port         Name               Status       Vlan       Duplex  Speed Type
Gi1/0/19                        notconnect   1            auto   auto 10/100/1000BaseTX
```

On a side note, IOS supports many valid abbreviations for the **interface GigabitEthernet 1/0/19** global configuration command, including the **int g1/0/19** command shown in the example.

Managing Switch Interface Configuration

This next section of the chapter examines a small number of additional interface subcommands, specifically the **description** and **shutdown** commands. The first gives you the ability to document facts about each interface, while the second gives you the means to disable and enable the interface administratively.

This section also explains more about the mechanisms of the IOS CLI for interfaces, with some discussion and examples of removing configuration from an interface using the **no** command.

The Description and Interface Range Commands

The **description** *text* interface subcommand lets you add a text description to the interface. For the text, you can use keyboard characters, including spaces, with Cisco switches typically supporting around 200 characters of description text. For instance, if you want to store some information in the switch about the interface and the device connected to it, you could document the details with the **description** interface subcommand.

Example 7-9 demonstrates the mechanics of the process with a simple description added to interface G1/0/1 from Figure 7-4. The example shows the configuration plus the output from some **show** commands that repeat the detail. Note that the **show interfaces status** command does not leave enough space for the entire **description** command's text, but the **show interface** command does.

Key Topic

Example 7-9 *Configuring and Verifying the* **description** *Command on Switch SW1*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface GigabitEthernet 1/0/1
SW1(config-if)# description Link to PC1, using autonegotiation
SW1(config-if)# ^Z
SW1# show interfaces g1/0/1 status

Port         Name               Status       Vlan     Duplex  Speed Type
Gi1/0/1      Link to PC1, using connected    1        a-full a-1000
10/100/1000BaseTX
SW1# show interfaces g1/0/1
GigabitEthernet1/0/1 is up, line protocol is up (connected)
  Hardware is Gigabit Ethernet, address is 4488.165a.f201 (bia 4488.165a.f201)
  Description: Link to PC1, using autonegotiation
! Lines omitted for brevity
SW1#
```

Example 7-10 demonstrates how to configure the same interface subcommand on multiple interfaces simultaneously, saving effort. The example uses the **description** command again but after the **interface range GigabitEthernet 1/0/2 - 10** command. The **interface range** command tells IOS to apply subsequent subcommands to multiple interfaces, in this case, Gi1/0/2 through Gi1/0/10.

Example 7-10 *Configuring Multiple Interfaces Using the* **interface range** *Command*

```
SW1# configure terminal
SW1(config)# interface range g1/0/2 - 10
SW1(config-if-range)# description Interface not in use
SW1(config-if-range)# ^Z
SW1#
```

IOS does not put the **interface range** command into the configuration. Instead, it acts as if you had typed the subcommands under every single interface in the specified range. Example 7-11 shows an excerpt from the **show running-config** command, listing the configuration of interfaces G1/0/2 –3 from the configuration in Example 7-10. The example shows the same **description** command on both interfaces; to save space, the example does not bother to show all interfaces G1/0/2 through G1/0/10.

Example 7-11 *How IOS Expands the Subcommands Typed After* **interface range**

```
SW1# show running-config
! Lines omitted for brevity
interface GigabitEthernet1/0/2
 description Interface not in use
!
interface GigabitEthernet1/0/3
 description Interface not in use
! Lines omitted for brevity
SW1# show interfaces description
Interface                      Status         Protocol Description
Vl1                            up             up
Gi0/0                          admin down     down
Gi1/0/1                        up             up       Link to PC1, using
autonegotiation
Gi1/0/2                        down           down     Interface not in use
Gi1/0/3                        down           down     Interface not in use
! Lines omitted for brevity
```

Administratively Controlling Interface State with shutdown

As you might imagine, network engineers need a way to enable and disable an interface using a command. In an odd turn of phrase, Cisco switches use the **shutdown** command to disable an interface and the **no shutdown** command to enable an interface. While the **no shutdown** command might seem like an odd command to enable an interface at first, you will use this command a lot in the lab, and it will become second nature. (Most people use the abbreviations **shut** and **no shut**.)

Example 7-12 shows an example of disabling an interface using the **shutdown** subcommand. In this case, switch SW1 has a working interface G1/0/1. The user connects to the switch console and disables the interface. IOS generates a log message each time an interface fails or recovers, and log messages appear at the console, as shown in the example.

Example 7-12 *Administratively Disabling an Interface with* **shutdown**

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface GigabitEthernet 1/0/1
SW1(config-if)# shutdown
*Oct  6 16:33:14.911: %LINK-5-CHANGED: Interface GigabitEthernet1/0/1, changed state
to administratively down
*Oct  6 16:33:15.911: %LINEPROTO-5-UPDOWN: Line protocol on Interface
GigabitEthernet1/0/1, changed state to down
SW1(config-if)#
```

To bring the interface back up again, you must follow the same process but use the **no shutdown** command instead.

Before leaving the simple but oddly named **shutdown/no shutdown** commands, examine the new status codes in the output in Example 7-13. The **show interfaces status** command lists one line of output per interface and, when shut down, lists the interface status as "disabled." That makes logical sense to most people. The **show interfaces** command (without the **status** keyword) lists many lines of output per interface, giving a much more detailed picture of interface status and statistics. With that command, the interface status comes in two parts, with one part using the phrase "administratively down," matching the highlighted log message in Example 7-12.

Example 7-13 *Interface Status When Configured with the* **shutdown** *Command*

```
SW1# show interfaces g1/0/1 status

Port         Name               Status       Vlan     Duplex  Speed Type
Gi1/0/1      Link to PC1, using disabled     1          auto   auto
10/100/1000BaseTX

SW1# show interfaces g1/0/1
GigabitEthernet1/0/1 is administratively down, line protocol is down (disabled)
  Hardware is Gigabit Ethernet, address is 4488.165a.f201 (bia 4488.165a.f201)
  Description: Link to PC1, using autonegotiation
! Lines omitted for brevity
```

Removing Configuration with the no Command

The chapters in Part II of this book have two broad goals: to help you learn some specific topics about LAN switches and to also learn about how to use the switch CLI. Some examples have more to do with learning about the CLI, which is the case for the examples that follow.

For any IOS configuration command that you might configure, you also need to consider this question: How can you remove that configuration? With some IOS configuration commands (but not all), you can revert to the default setting by issuing a **no** version of the command. What does that mean? Let me give you a few examples:

- If you configured **speed 1000** on an interface, the **no speed** command on that same interface reverts to the default speed setting (which happens to be **speed auto**).
- Similarly, if you configured an earlier **duplex half** or **duplex full** command, the **no duplex** command in interface mode for the same interface reverts the configuration to the default **duplex auto**.

- If you configured a **description** command with some text, to go back to the default state of having no **description** command for that interface, you can use the **no description** command when in interface configuration mode for that same interface.

Example 7-14 shows a sample interface configuration with four interface subcommands configured. Example 7-15 demonstrates the configuration's removal using various **no** commands.

Example 7-14 *Existing Configuration on Switch SW1 Interface G1/0/21*

```
SW1# show running-config interface g1/0/21
Building configuration...

Current configuration : 96 bytes
!
interface GigabitEthernet1/0/21
 description link to switch SW2
 speed 1000
 duplex full
 shutdown
end
```

Example 7-15 *Removing Various Configuration Settings Using the* **no** *Command*

```
SW1# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
SW1(config)# interface gigabitethernet 1/0/21
SW1(config-if)# no speed
SW1(config-if)# no duplex
SW1(config-if)# no description
SW1(config-if)# no shutdown
SW1(config-if)# ^Z
SW1#
SW1# show running-config interface g1/0/21
Building configuration...

Current configuration : 39 bytes
!
interface GigabitEthernet1/0/21
end
SW1#
```

In particular, interface g1/0/21 has no interface subcommands at the bottom of Example 7-15.

NOTE The **show running-config** and **show startup-config** commands typically do not display default configuration settings. Hence, the absence of interface subcommands under an interface means that all possible subcommands use default values. You can see the configured and default settings using the **show running-config all** command. (Be warned, the **show running-config all** command lists many commands, about ten times the output of the **show running-config** command.)

Alternatively, if the goal is to revert all interface subcommands to their default settings, later IOS versions provide a single command: the **default interface** *interface-id* global configuration command. For instance, if switch SW1 had the configuration shown in Example 7-14 again, the global configuration command **default interface g1/0/21** would accomplish the same result as the list of **no** commands seen in Example 7-15, resulting in all default configuration on the interface.

Analyzing Switch Interface Status and Statistics

This final major section of the chapter examines how to verify the interfaces work correctly. This section also looks at those more unusual cases in which the interface is working but not working well, as revealed by different interface status codes and statistics.

Interface Status Codes

Cisco switches use two different sets of interface status codes. The switch **show interfaces** and **show interfaces description** commands list a two-code status named the *line status* and *protocol status*. Together, these two status values identify the state of the interface. Generally, the line status refers to whether the Layer 1 standard works, while the protocol status refers to whether Layer 2 works.

NOTE This book refers to these two status codes in shorthand by just listing the two codes with a slash between them, such as *up/up* or *down/down*.

Other commands, like the **show interfaces status** command, use a single-code interface status. The single-word status words correlate to different combinations of the two-code interface status codes, as seen in Table 7-2. For example, the **show interfaces status** command lists the *connected* state for working interfaces, while the **show interfaces** and **show interfaces description** commands list an *up/up* state.

Key Topic

Table 7-2 LAN Switch Interface Status Codes

Line Status	Protocol Status	Interface Status	Typical Root Cause
administratively down	down	disabled	The **shutdown** command is configured on the interface.
down	down	notconnect	No cable; bad cable; wrong cable pinouts with MDIX disabled; speed mismatch; the neighboring device is (a) powered off, (b) **shutdown**, or (c) error disabled.
up	down	notconnect	Not expected on LAN switch physical interfaces.
down	down (err-disabled)	err-disabled	Port security (or other feature) has disabled the local interface.
up	up	connected	The interface is working.

Examining the notconnect state for a moment, note that this state has many causes. For example, this state includes the more obvious cases, like a missing or broken cable. Some examples of the root causes of cabling problems that result in a notconnect state include the following:

- The installation of any equipment that uses electricity, even non-IT equipment, can interfere with the transmission on the cabling and link fail.
- The cable could be damaged, for example, if it lies under the carpet. The electrical signal can eventually degrade if the user's chair squashes the cable.
- Although optical cables do not suffer from electromagnetic interference (EMI), someone can try to be helpful and move a fiber-optic cable out of the way—by bending it too much. A bend into too tight a shape can prevent the cable from transmitting bits (called *macrobending*).

The Duplex Mismatch Issue

You might think the up/up (connected) state means everything works. Indeed, you want your switch interfaces in that state; however, a variety of problems can occur on a working interface in that state, one being a *duplex mismatch*, as discussed next.

If the devices on the ends of a link use the same physical layer standard with a working cable, the interfaces reach the up/up (connected) state. However, the endpoints can also have opposite duplex settings (a duplex mismatch), with full on one side and half on the other. The interfaces remain up/up and data flows; however, the device that uses half duplex experiences unnecessary errors and retransmissions.

You should remember the recommendation by now: Avoid problems like duplex mismatches by using autonegotiation on both devices on each link. However, because the CCNA exam topics have mentioned the duplex mismatch problem for most of its history, take a closer look at how a duplex mismatch can occur.

Figure 7-7 shows a classic case that results in a duplex mismatch with some Cisco switch ports. Some Cisco switch ports disable autonegotiation on interfaces configured with both a specific speed and duplex setting. The device that uses autonegotiation uses parallel detection rules to discover the speed and then uses a default duplex based on that speed, but that default duplex setting may not match the device on the other end.

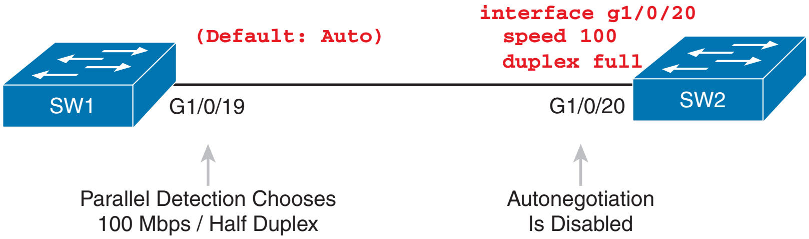


Figure 7-7 *Classic Case for Duplex Mismatch Between Switches*

Figure 7-7 shows that scenario, with switch SW2's G1/0/20 interface configured with **speed 100** and **duplex full**, disabling autonegotiation. The logic runs like this:

1. Switch SW2 sets speed 100 and duplex full.
2. SW2 disables autonegotiation FLP messages.

3. Because it receives no FLP messages, switch SW1 moves on to use autonegotiation parallel detection logic:
    a. SW1 senses that SW2 uses 100 Mbps speed.
    b. SW1 chooses to use half duplex based on the default table (half duplex if the speed is 10 or 100 Mbps; otherwise, full duplex).

Finding a duplex mismatch can be much more difficult than finding a speed mismatch because speed mismatches result in a failed link, but a duplex mismatch does not. In the duplex mismatch case as shown in Figure 7-7, *if the duplex settings do not match on the ends of an Ethernet segment, the switch interface will still be in a connected state.*

To identify duplex mismatch problems, you have to check the duplex setting on each end of the link to see if the values mismatch. You can also watch for incrementing collision and late collision counters, as explained in the next section.

NOTE Some Cisco switch ports do not disable autonegotiation when configured with both **speed** and **duplex** as shown in Figure 7-7. It appears that switch ports that support Power over Ethernet (PoE) do not disable autonegotiation, while ports that do not support PoE do disable autonegotiation, as shown in the figure. However, I found these facts only from experimentation, and not from any Cisco documentation, so be cautious and test if you decide to configure both speed and duplex. Also, the fact that some ports disable autonegotiation when configured with both speed and duplex, but some do not, provides yet another reason to simply use autonegotiation on both ends of the link.

On switch ports that continue using autonegotiation, even after you configure the speed and duplex commands, a duplex mismatch should not occur, because autonegotiation works. Working through the Figure 7-7 example again, but assuming a port that does not disable autonegotiation, consider this sequence:

1. Switch SW2 sets speed 100 and duplex full.
2. SW2 continues to send autonegotiation FLP messages, declaring it can support only 100 Mbps and full duplex.
3. SW1, using autonegotiation, receives SW2's FLPs, and agrees to use 100 Mbps and full duplex—avoiding the duplex mismatch.

Common Layer 1 Problems on Working Interfaces

When the interface reaches the connected (up/up) state, the switch considers the interface to be working. The switch, of course, tries to use the interface, and at the same time, the switch keeps various interface counters. These interface counters can help identify problems that can occur even though the interface reaches a connected state, like issues related to the just-completed duplex mismatch problem. This section explains some of the related concepts and a few of the most common problems.

The receiving device might receive a frame whose bits have changed values whenever the physical transmission has problems. These frames do not pass the error detection logic as implemented in the FCS field in the Ethernet trailer, as covered in Chapter 2. The receiving device discards the frame and counts it as an *input error*. Cisco switches list this error as a CRC error, as highlighted in Example 7-16. (Cyclic redundancy check [CRC] is a term related to how the frame check sequence [FCS] math detects an error.)

Example 7-16 *Interface Counters for Layer 1 Problems*

```
SW1# show interfaces gi1/0/1
! lines omitted for brevity
     Received 3943 broadcasts (3941 multicasts)
     0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored
     0 watchdog, 3941 multicast, 0 pause input
     0 input packets with dribble condition detected
     18843 packets output, 1726956 bytes, 0 underruns
     Output 10 broadcasts (16378 multicasts)
     0 output errors, 0 collisions, 3 interface resets
     0 unknown protocol drops
     0 babbles, 0 late collision, 0 deferred
     0 lost carrier, 0 no carrier, 0 pause output
     0 output buffer failures, 0 output buffers swapped out
```

The numbers of input and CRC errors are just a few of the counters in the output of the **show interfaces** command. The challenge is to decide which counters you need to think about, which ones show that a problem is happening, and which ones are normal and of no concern.

The example highlights several counters as examples so that you can start to understand which ones point to problems and which ones are just counting everyday events that are not problems. The following list shows a short description of each highlighted counter in the order shown in the example:

Runts: Frames that did not meet the minimum frame size requirement (64 bytes, including the 18-byte destination MAC, source MAC, type, and FCS). Collisions can cause it.

Giants: Frames that exceed the maximum frame size requirement (default 1518 bytes, including the 18-byte destination MAC, source MAC, type, and FCS).

Input Errors: A total of many counters, including runts, giants, no buffer, CRC, frame, overrun, and ignored counts.

CRC: Received frames that did not pass the FCS math; can be caused by collisions.

Frame: Received frames that have an illegal format, for example, ending with a partial byte, can be caused by collisions.

Packets Output: Total number of packets (frames) forwarded out the interface.

Output Errors: Total number of packets (frames) that the switch port tried to transmit but for which some problem occurred.

Collisions: Counter of all collisions that occur when the interface is transmitting a frame.

Late Collisions: The subset of all collisions that happen after the 64th byte of the frame has been transmitted. (In a properly working Ethernet LAN, collisions should occur within the first 64 bytes; **late collisions** today often point to a duplex mismatch.)

Note that many of these counters increment on a properly working interface that uses the CSMA/CD process to implement half duplex. So, a switch interface with an increasing collision counter might not have a problem. However, one problem, called late collisions, points to the classic duplex mismatch problem.

If a LAN design follows cabling guidelines, all collisions should occur by the end of the 64th byte of any frame. When a half-duplex switch interface has already sent 64 bytes of a frame and receives a frame on that same interface, the switch senses a collision. In this case, the collision is a late collision, and the switch increments the late collision counter in addition to the usual CSMA/CD actions to send a jam signal, wait a random time, and try again.

With a duplex mismatch, like the mismatch between SW1 and SW2 in Figure 7-7, the half-duplex interface will likely see the late collisions counter increment. Why? The half-duplex interface sends a frame (SW1), but the full-duplex neighbor (SW2) sends at any time, even after the 64th byte of the frame sent by the half-duplex switch. So, just keep repeating the **show interfaces** command, and if you see the late collisions counter incrementing on a half-duplex interface, you might have a duplex mismatch problem.

A working interface (in an up/up state) can also suffer from issues related to the physical cabling. The cabling problems might not be bad enough to cause a complete failure, but the transmission failures result in some frames failing to pass successfully over the cable. For example, excessive interference on the cable can cause the various input error counters to keep growing, especially the CRC counter. In particular, if the CRC errors grow, but the collision counters do not, the problem might simply be interference on the cable.

Chapter Review

One key to doing well on the exams is to perform repetitive spaced review sessions. Review this chapter's material using either the tools in the book or interactive tools for the same material found on the book's companion website. Refer to the "Your Study Plan" element section titled "Step 2: Build Your Study Habits Around the Chapter" for more details. Table 7-3 outlines the key review elements and where you can find them. To better track your study progress, record when you completed these activities in the second column.

Table 7-3 Chapter Review Tracking

Review Element	Review Date(s)	Resource Used
Review key topics		Book, website
Review key terms		Book, website
Answer DIKTA questions		Book, PTP
Review command tables		Book
Review memory tables		Book, website
Do labs		Sim Lite, blog

Review All the Key Topics

Key Topic

Table 7-4 Key Topics for Chapter 7

Key Topic Element	Description	Page Number
List	IEEE autonegotiation rules when both link partners participate	162
List	IEEE autonegotiation rules when only one link partner participates (parallel detection)	163
Example 7-1	Confirming All Default Settings on the Switch Interfaces	166
Example 7-3	Interpreting Autonegotiation Clues Before/After Completion	167
Example 7-4	**show interfaces** Command Autonegotiation Clues	168
Example 7-5	Configuring Speed and Duplex on Both Ends of a Link	169
Example 7-6	**show interfaces status** Without Using Autonegotiation	170
Example 7-9	Configuring and Verifying the **description** Command on Switch SW1	172
Example 7-12	Administratively Disabling an Interface with **shutdown**	174
Example 7-13	Interface Status When Configured with the **shutdown** Command	174
Table 7-2	Two types of interface state terms and their meanings	176
List	Explanations of different error statistics on switch interfaces	179

Key Terms You Should Know

10/100, 10/100/1000, auto-MDIX, autonegotiation, duplex mismatch, full duplex, half duplex, late collisions, parallel detection

Do Labs

The Sim Lite software is a version of Pearson's full simulator learning product with a subset of the labs, included free with this book. The subset of labs mostly relates to this part. Take the time to try some of the labs. As always, also check the author's blog site pages for configuration exercises (Config Labs) at https://www.certskills.com.

Command References

Tables 7-5 and 7-6 list configuration and verification commands used in this chapter. As an easy review exercise, cover the left column in a table, read the right column, and try to recall the command without looking. Then repeat the exercise, covering the right column, and try to recall what the command does.

Table 7-5 Switch Interface Configuration

Command	Mode/Purpose/Description
interface *type port-number*	Changes context to interface mode. The type is typically Fast Ethernet or Gigabit Ethernet. The possible port numbers vary depending on the model of switch—for example, Fa0/1, Fa0/2, and so on.
interface range *type port-number - end-port-number*	Changes the context to interface mode for a range of consecutively numbered interfaces. The subcommands that follow then apply to all interfaces in the range.
shutdown | **no shutdown**	Interface mode. Disables or enables the interface, respectively.
speed {**10** | **100** | **1000** | **auto**}	Interface mode. Manually sets the speed to the listed speed or, with the auto setting, automatically negotiates the speed.
duplex {**auto** | **full** | **half**}	Interface mode. Manually sets the duplex to half or full, or to autonegotiate the duplex setting.
description *text*	Interface mode. Lists any information text the engineer wants to track for the interface, such as the expected device on the other end of the cable.
no duplex **no speed** **no description**	Reverts to the default setting for each interface subcommand of **speed auto**, **duplex auto**, and the absence of a **description** command.
default interface *interface-id*	Reverts to the default setting for all interface subcommands on an interface.
[no] mdix auto	Interface subcommand. Enables (**auto mdix**) or disables (**no auto mdix**) the auto-MDIX feature.

Table 7-6 Chapter 7 EXEC Command Reference

Command	Purpose
show running-config	Lists the currently used configuration while omitting most default settings
show running-config interface *type number*	Displays the running-configuration excerpt of the listed interface and its subcommands only
show running-config all	Displays the running-configuration while including all default settings
show interfaces [*type number*] **status**	Lists one output line per interface (or for only the listed interface if included), noting the description, operating state, and settings for duplex and speed on each interface
show interfaces [*type number*]	Lists detailed status and statistical information about all interfaces (or the listed interface only)
show interfaces description	Displays one line of information per interface, with a two-item status (similar to the **show interfaces** command status), and includes any description configured on the interfaces

This page intentionally left blank

Part II Review

Keep track of your part review progress with the checklist shown in Table P2-1. Details on each task follow the table.

Table P2-1 Part II Part Review Checklist

Activity	1st Date Completed	2nd Date Completed
Repeat All DIKTA Questions		
Answer Part Review Questions		
Review Key Topics		
Do Labs		
Review Appendix L on the Companion Website		
Watch video		
Use Per-Chapter Interactive Review		

Repeat All DIKTA Questions

For this task, answer the "Do I Know This Already?" questions again for the chapters in this part of the book, using the PTP software.

Answer Part Review Questions

For this task, answer the Part Review questions for this part of the book, using the PTP software.

Review Key Topics

Review all key topics in all chapters in this part, either by browsing the chapters or by using the Key Topics application on the companion website.

Labs

Depending on your chosen lab tool, here are some suggestions for what to do in lab:

Pearson Network Simulator: If you use the full Pearson ICND1 or CCNA simulator, focus more on the configuration scenario and troubleshooting scenario labs associated with the topics in this part of the book. These types of labs include a larger set of topics and work well as Part Review activities. (See the Introduction for some details about how to find which labs are about topics in this part of the book.)

Blog: Config Labs: The author's blog includes a series of configuration-focused labs that you can do on paper or with Cisco Packet Tracer in about 15 minutes. To find them, open https://www.certskills.com and look under the Labs menu item.