Chapter 17

SNMP, FTP, and TFTP

This chapter covers the following exam topics:

4.0 Infrastructure Services

4.4 Explain the function of SNMP in network operations

4.9 Describe the capabilities and functions of TFTP/FTP in the network

To close this part of the book about a variety of IP Services, this chapter focuses on two topics of managing routers and switches as an end to themselves.

Network engineers have used Simple Network Management Protocol (SNMP) as the main network management protocol for decades. As per the associated exam topic, the first major section focuses on SNMP concepts rather than configuration, including how managed devices—SNMP agents—can be interrogated by network management systems—SNMP clients—to find the current status of each device.

The second major section of the chapter examines the mechanics of transferring files with the FTP and TFTP protocols. The first topic of this section focuses on a few practical uses of TFTP and FTP when working with Cisco routers to upgrade the IOS. Armed with that practical knowledge, you then look at the protocol details of both FTP and TFTP in the rest of the section.

“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 17-1 “Do I Know This Already?” Foundation Topics Section-to-Question Mapping

Foundation Topics Section	Questions
Simple Network Management Protocol	1–3
FTP and TFTP	4–6

1. A Network Management Station (NMS) is using SNMP to manage some Cisco routers and switches with SNMPv2c. Which of the following answers most accurately describes how the SNMP agent on a router authenticates any SNMP Get requests received from the NMS?

   a. Using a username and hashed version of a password

   b. Using either the read-write or read-only community string

   c. Using only the read-write community string

   d. Using only the read-only community string

2. Which of the following SNMP messages are typically sent by an SNMP agent? (Choose two answers.)

   a. Trap

   b. Get Request

   c. Inform

   d. Set Request

3. Which SNMP verbs were added for SNMP Version 2 to improve the efficiency of retrieving lists of related MIB variables?

a. GetBulk

b. Get

c. GetNext

d. Inform

4. An FTP client connects to an FTP server using active mode and retrieves a copy of a file from the server. Which of the answers describes a TCP connection initiated by the FTP client?

a. The FTP control connection only

b. The FTP data connection only

c. Both the FTP data and control connections

d. Neither the FTP data nor control connections

5. Which of the following functions are supported by FTP but not by TFTP? (Choose two answers.)

a. Transferring files from client to server

b. Changing the current directory on the server

c. Transferring files from server to client

d. Listing directory contents of a server's directory

6. In an IOS XE router, the **pwd** command output lists directory "bootflash:". Which answers describe what you would expect to see in the output of the **show bootflash:** and **dir** commands? (Choose two answers.)

a. The **dir** command lists all files in file system bootflash:, including files in subdirectories.

b. The **show bootflash:** command lists all files in file system bootflash:, including files in subdirectories.

c. The **dir** command lists files in the root of bootflash:, but not files in subdirectories.

d. The **show bootflash:** command lists files in the root of bootflash:, but not files in subdirectories.

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

1 B

2 A, C

3 A

4 A

5 B, D

6 B, C

Foundation Topics

Simple Network Management Protocol

In 1988, RFC 1065, "Structure and Identification of Management Information for TCP/IP-based Internets," was published. SNMP uses a model of breaking down the information about devices on a TCP/IP-based network—configuration settings, status information, counters, and so on—into a database of variables. Management software could then collect those variables to monitor and manage the IP-based network. After all, the elements of any IP-based machines would have commonalities. For example, a PC, a network printer, and a router would all have commonalities such as interfaces, IP addresses, and buffers. Why not create a standardized database of these variables and a simple system for monitoring and managing them? This idea was brilliant, caught on, and became what we know today as **Simple Network Management Protocol (SNMP)**.

In this major section of the chapter, we now turn our attention to SNMP by looking at the major concepts along with the two common versions used today: **SNMPv2c** and **SNMPv3**.

SNMP defines request and response messages for communication between **SNMP managers** and **SNMP agents**. An SNMP manager is a network management application running on a PC or server, with that host typically called a **Network Management System (NMS)**. Many SNMP agents exist in the network, one per device that is managed. The SNMP agent is software running inside each device (router, switch, and so on), with knowledge of all the variables on that device that describe the device's configuration, status, and counters. The SNMP manager uses SNMP protocols to communicate with each SNMP agent.

Each agent keeps a database of variables that comprise the parameters, status, and counters for the device's operations. This database, called the **Management Information Base (MIB)**, has some core elements in common across most networking devices. It also has many variables unique to that type of device—for instance, router MIBs will include variables not needed on switch MIBs, and vice versa. (For perspective, I checked a router when writing this section and found a little over 7000 MIB variables on a router.)

Figure 17-1 connects a few of these ideas and terms. First, many companies sell SNMP management products. For example, the Cisco Catalyst Center (formerly Cisco DNA Center) controller (www.cisco.com/go/catalystcenter) uses SNMP (and other protocols) to manage networks. IOS on routers and switches include an SNMP agent with built-in MIB.

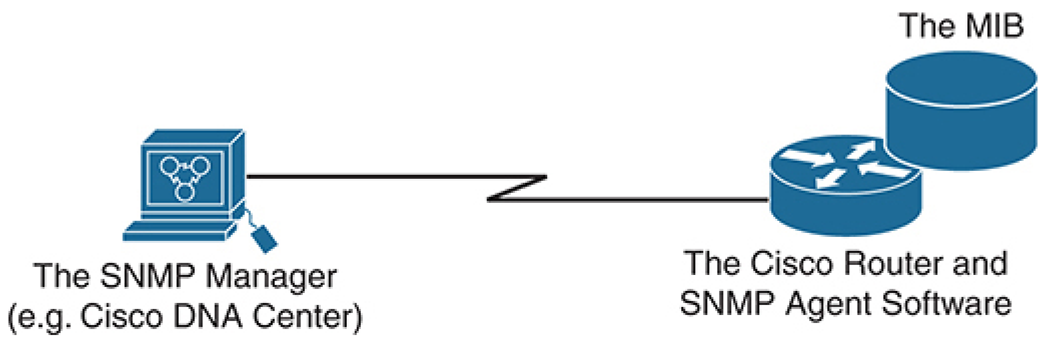


Figure 17-1 *Elements of Simple Network Management Protocol*

SNMP Variable Reading and Writing: SNMP Get and Set

The NMS typically polls the SNMP agent on each device. The NMS can notify the human user in front of the PC or send emails, texts, and so on to notify the network operations staff of any issues identified by the data found by polling the devices. You can even reconfigure the device through these SNMP variables in the MIB if you permit this level of control.

Specifically, the NMS uses the **SNMP Get** request message to ask for information from an agent. The NMS sends an **SNMP Set** request message to write variables on the SNMP agent to configure the device. These messages come in pairs, with, for instance, a Get Request asking the agent for the contents of a variable, and the Get Response supplying that information. Figure 17-2 shows an example of a typical flow, with the NMS using an SNMP Get to ask for the MIB variable that describes the status of a particular router interface.

Key Topic

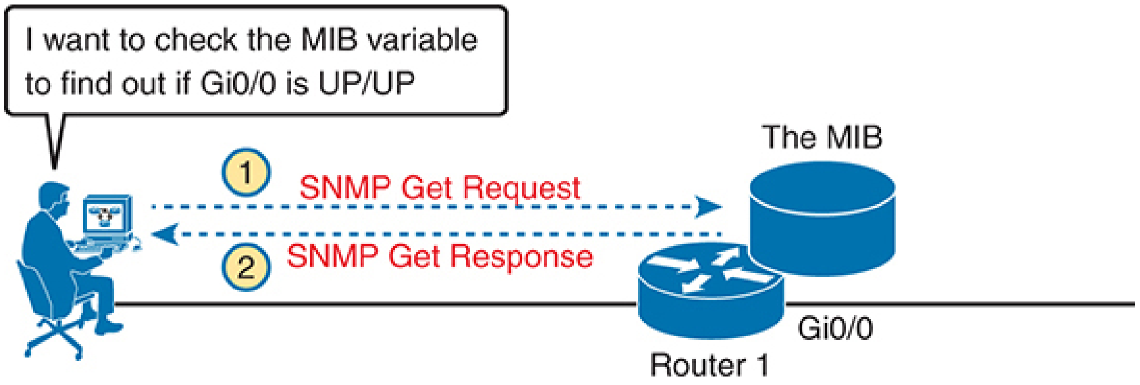

I want to check the MIB variable to find out if Gi0/0 is UP/UP
1
SNMP Get Request
2
SNMP Get Response
The MIB
Gi0/0
Router 1

Figure 17-2 *SNMP Get Request and Get Response Message Flow*

The original SNMP Version 1 (SNMPv1) standard defined two types of Get requests: Get and GetNext. The Get request lists one or more specific variable names, with the SNMP agent sending back a Get response that lists the values of those specific variables. However, the Get response requires the listing of a specific variable, which requires some processing by the NMS.

The GetNext request takes a slightly different approach that reduces some processing at the NMS. GetNext also asks for the contents of variables; however, a GetNext request asks for the next variable in the agent’s MIB after the listed variable. So, the NMS can do the work to find the specific name of one variable, and then issue a series of GetNext requests to get the next instance of that variable, and the next, and so on.

For instance, imagine a router has many interfaces, as usual. The router’s MIB has variables for each interface, for instance, for interface MAC and IP addresses. First, the NMS identifies the specific variable names for the first interface and uses a Get request on those variables. However, to get those same variables for the second interface, GetNext can refer to the names of the variables for the first interface, without having to do the processing to derive the specific variable names for the second interface. It reduces the overhead to traverse and retrieve variables in a list.

SNMP Version 2 (SNMPv2) added the GetBulk request as an additional improvement over GetNext. While one GetNext requests the next variable in a list or table, GetBulk can request more than one successive variable. For instance, to retrieve data for eight router interfaces, the NMS could begin with one Get and then be more efficient with seven more GetNext requests, for each additional interface. One GetBulk request could ask for the MAC and IP address of the next eight interfaces on a router, again making the process more efficient.

SNMP permits much flexibility in how you monitor variables in the MIB. A network administrator gathers and stores statistics over time using the NMS. With the stored data, the NMS can analyze various statistical facts such as averages, minimums, and maximums. To be proactive, administrators can set thresholds for certain key variables, telling the NMS to send a notification (email, text, and so on) when a threshold is passed.

SNMP Notifications: Traps and Informs

In addition to asking for information with Get commands and setting variables on agents with the Set command, SNMP agents can initiate communications to the NMS. These messages, generally called *notifications*, use two specific SNMP messages: Trap and Inform. Agents send an SNMP Trap or **SNMP Inform** message to the NMS to list the value of certain MIB variables when those variables reach a certain state.

As an example of a Trap, suppose that Router 1's G0/0 interface fails, as shown at step 1 of Figure 17-3. With Traps configured, the router would send an **SNMP Trap** message to the NMS, with that Trap message noting the down state of the G0/0 interface. Then, the NMS software can send a text message to the network support staff, pop up a window on the NMS screen, change the color of the correct router icon to red on the graphical interface, and so on.

Key Topic

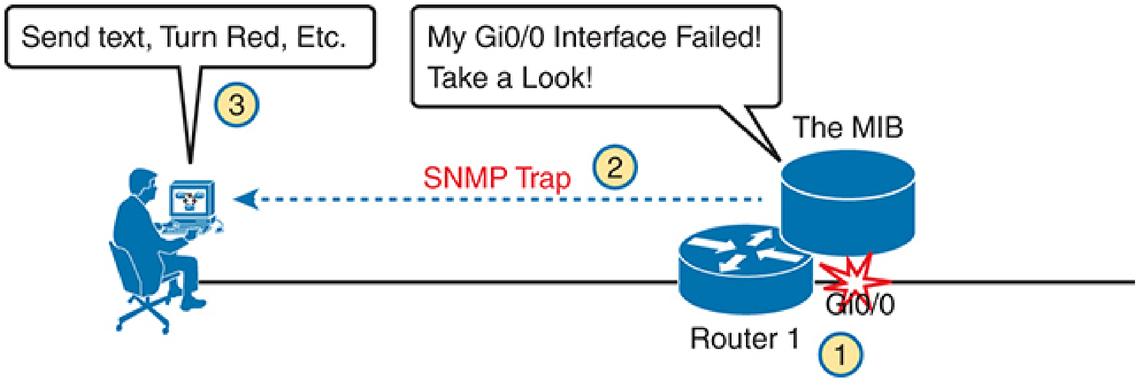

Send text, Turn Red, Etc.
3
My Gi0/0 Interface Failed! Take a Look!
The MIB
SNMP Trap
2
Gi0/0
Router 1
1

Figure 17-3 *SNMP Trap Notification Process*

SNMP Trap and Inform messages have the exact same purpose but differ in the protocol mechanisms. SNMP Traps, available since the first version of SNMP (SNMPv1), use a fire-and-forget process. The SNMP agent sends the Trap to the IP address of the NMS, with UDP as the transport protocol as with all SNMP messages, and with no application layer error recovery. If the Trap arrives, great; if it is lost in transit, it is lost.

Inform messages inform the NMS but with reliability added. Added to the protocol with SNMPv2, Informs also use UDP, but add application layer reliability. The NMS must acknowledge receipt of the Inform with an SNMP Response message, or the SNMP agent will time out and resend the Inform.

Note that Traps and Informs both have a useful role today, and Traps are still frequently used. Both inform the NMS. Traps use less overhead on the agent, while Informs improve reliability of the messages but require a little more overhead effort.

The Management Information Base

Every SNMP agent has its own Management Information Base. The MIB defines variables whose values are set and updated by the agent. The MIB variables on the devices in the network enable the management software to monitor/control the network device.

Each agent organizes its MIB in a hierarchical structure, with the individual variables called an *object ID* (OID). Each node in the tree can be described based on the tree structure sequence, either by name or by number. Figure 17-4 shows a small part of the tree structure of an MIB that happens to be part of the Cisco-proprietary part of the MIB.

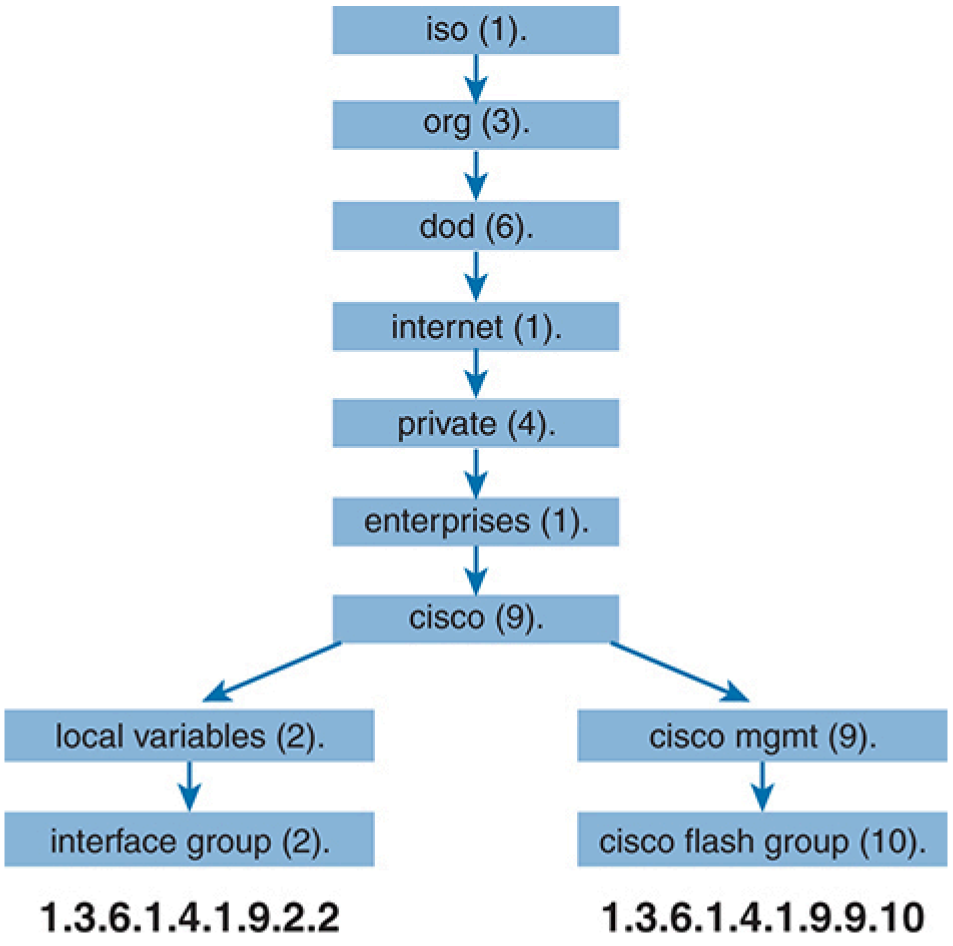


Figure 17-4 *Management Information Base (MIB)*

Working directly with an MIB, with long variable names and numbers, can be a bit of a challenge, so NMS software typically hides the complexity of the MIB variable numbering and names. However, to get a sense for the variable names, Figure 17-4 shows the tree structure for two variables, with the variable names being the long string of numbers shown at the bottom of the figure. Working with those numbers and the tree structure can be difficult at best. As a result, most people manage their networks using an NMS. For perspective, you could use an SNMP manager and type **MIB variable 1.3.6.1.4.1.9.2.1.58.0** and click a button to get that variable, to see the current CPU usage percentage from a Cisco router. However, most users of an NMS would much prefer to ignore those details and have a simple graphical interface to ask for the same information, never having to know that 1.3.6.1.4.9.2.1.58.0 represents the router CPU utilization MIB variable.

For an SNMP manager to manage each agent, the SNMP manager must have the same MIBs used by the various SNMP agents installed for proper SNMP operation. Each agent comes with a set of standard and vendor-proprietary MIBs built into the product. Expecting that, SNMP manager products also preload the most common MIBs in anticipation of managing those agents. Figure 17-5 shows the concept, with an SNMP manager on the left. It also shows a Cisco and non-Cisco switch on the right. The switches both support the standard MIB-II MIB plus a larger proprietary MIB built by each switch vendor.

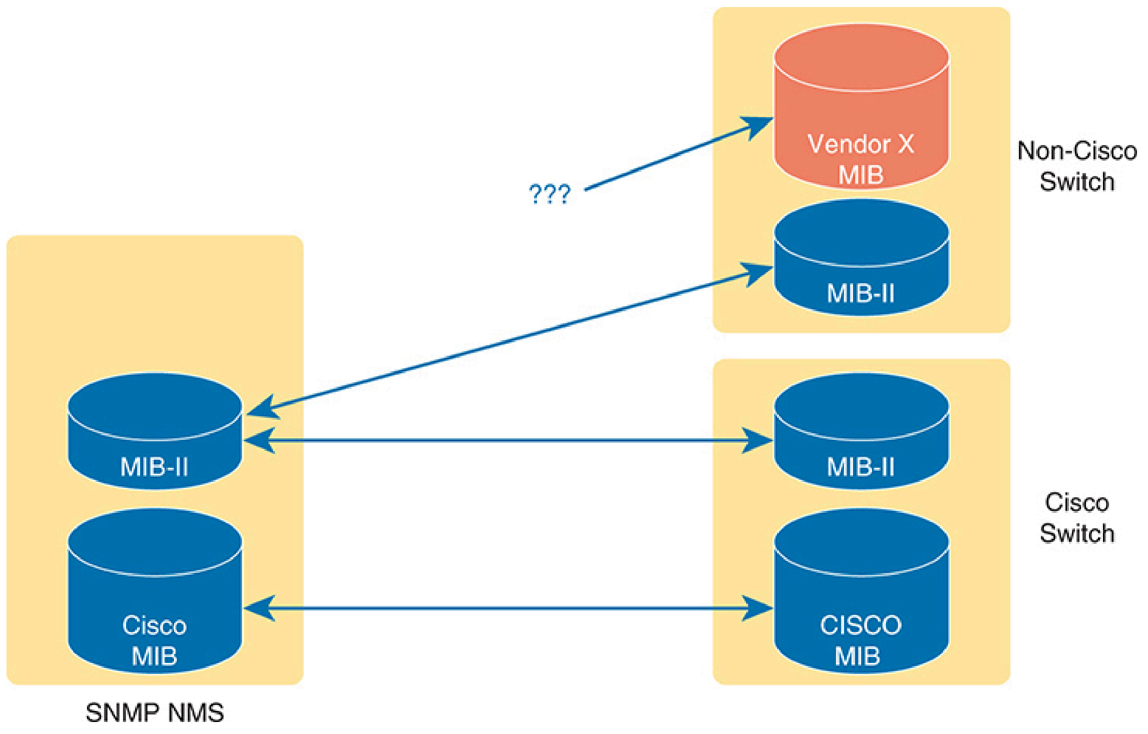


Figure 17-5 *Proprietary MIB Concepts*

The Cisco NMS comes preloaded with the MIBs for Cisco products, but not for all non-Cisco products. The arrow lines show the MIBs that match between the devices (agents) and the NMS. For those MIB variables, the NMS can issue Get requests to the Cisco switch. The NMS will also understand those variables in Trap or Inform messages. Likewise, the NMS can support Gets, Sets, Traps, and Informs using the MIB II in the non-Cisco switch. However, without the non-Cisco vendor's MIB loaded into the NMS, the NMS cannot ask for data from the non-Cisco vendor's MIB, and it will not understand Traps and Informs that use variables from that MIB. The solution? Download and install the other vendor's MIB into the NMS.

Securing SNMP

SNMP supports a few security mechanisms, depending in part on the particular version. This section works through the options.

First, one strong method to secure SNMP is to use ACLs to limit SNMP messages to those from known servers only. SNMP agents on Cisco routers and switches support SNMP messages that flow in both IPv4 and IPv6 packets. The SNMP agent can configure an IPv4 ACL to filter incoming SNMP messages that arrive in IPv4 packets and an IPv6 ACL to filter SNMP messages that arrive in IPv6 packets.

Using an IPv4 and IPv6 ACL to secure an agent makes good sense. The only hosts that should be sending SNMP messages to the SNMP agent in a router or switch are the NMS hosts. Those NMS hosts seldom move and their IP addresses should be well known to the networking staff. It makes good sense to configure an ACL that permits packets sourced from the IP addresses of all NMS hosts, but no others.

As for the SNMP protocol messages, all versions of SNMP support a basic clear-text password mechanism, although none of those versions refer to the mechanism as using a password. SNMP Version 3 (SNMPv3) adds more modern security as well.

SNMPv1 defined clear-text passwords called **SNMP communities**. Basically, both the SNMP agent and the SNMP manager need prior knowledge of the same SNMP community value (called a *community string*). The SNMP Get messages and the Set message include the appropriate community string value, in clear text. If the NMS sends a Get or Set with the correct community string, as configured on the SNMP agent, the agent processes the message.

SNMPv1 defines both a read-only community and a read-write community. The **read-only (RO) community** allows Get messages, and the **read-write (RW) community** allows both reads and writes (Gets and Sets). Figure 17-6 shows the concepts. At steps 1 and 2, the agent is configured with particular RO and RW community strings, and the NMS configures the matching values. At step 3, the SNMP Get can flow with either community, but at step 4, the Set Request must use the RW community.

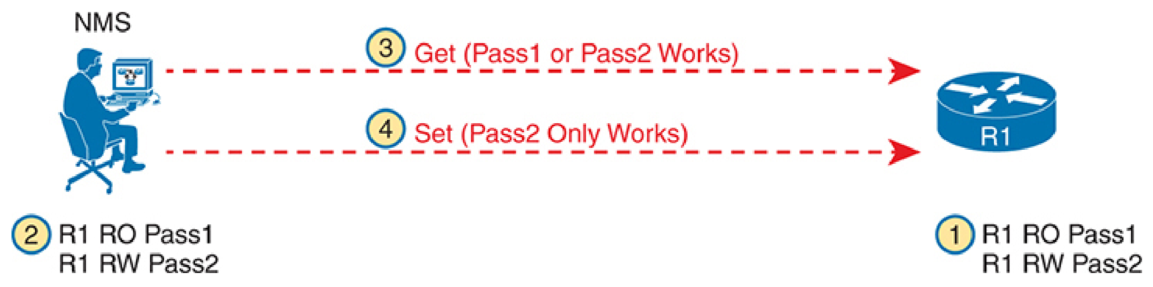


Figure 17-6 *RO and RW Communities with the Get and Set Commands*

SNMPv2 and the related Community-based SNMP Version 2 (SNMPv2c) added a wrinkle in naming but basically kept the same community security feature as SNMPv1 once the standards process completed. The original specifications for SNMPv2 did not include SNMPv1 communities; however, the marketplace still wanted communities, so an additional RFC added the SNMPv1 communities mechanism back to SNMPv2. This updated RFC, "Community-based SNMPv2," came to be known simply as SNMPv2c. Vendors (including Cisco) implemented SNMPv2c; however, security was still relatively weak.

SNMPv3 arrived with much celebration among network administrators. Finally, security had arrived with the powerful network management protocol. SNMPv3 does away with communities and replaces them with the following features:

Key Topic

- **Message integrity:** This mechanism, applied to all SNMPv3 messages, confirms whether or not each message has been changed during transit.
- **Authentication:** This optional feature adds authentication with both a username and password, with the password never sent as clear text. Instead, it uses a hashing method like many other modern authentication processes.
- **Encryption (privacy):** This optional feature encrypts the contents of SNMPv3 messages so that attackers who intercept the messages cannot read their contents.

Note

The CCNA 200-301 version 1.1 exam blueprint lists SNMP in one exam topic, with that exam topic reduced to "explain SNMP." Some previous versions of the exam required configuration. Refer to Appendix D, "Topics from Previous Editions," if you're interested in learning about SNMP configuration and verification.

FTP and TFTP

This final major section of the chapter focuses on two topics: File Transfer Protocol (**FTP**) and Trivial File Transfer Protocol (**TFTP**). Both exist as TCP/IP protocols defined in RFCs. Both use a client and server model, in which the client connects to a server and then the client can copy files to the server or from the server. Both exist as a myriad of implementations of both client and server code, from command-line clients to apps with graphical interfaces, using the respective FTP or TFTP protocols behind the scenes.

This section discusses FTP and TFTP with two branches. First, we take a practical view of the most common use of TFTP and FTP by network engineers while on the job: the job of updating IOS images. The process can make use of TFTP and FTP, so this section provides the basics. Second, we move on to talk about FTP and TFTP in a much broader sense, with details about each protocol, their capabilities, and what capabilities each provides to any user.

Managing Cisco IOS Images with FTP/TFTP

IOS exists as a file—a single file—that the router then loads into RAM to use as its operating system. To better understand the process, you must understand a few more details about how IOS works. In particular, you need to understand the **IOS file system (IFS)**, which defines how IOS stores files (including the IOS file). The **IOS image** upgrade process occurs by copying new IOS files into the router and then booting the router with that new IOS.

The IOS File System

Every OS creates file systems to store files. A computer needs some type of permanent storage, but it needs more than just a place to store bytes. The OS organizes the storage into a file system, which includes directories, a structure, and filenames, with the associated rules. By using a file system, the OS can keep data organized so the user and the applications can find the data later.

Every OS defines its own file system conventions. Windows OSs, for instance, use a left-leaning slash (\) in directory structures, like \Desktop\Applications. Linux and macOS use a right-leaning slash, for example, /Desktop. Each OS refers to physical disks slightly differently as well, and IOS is no different.

As for the physical storage, Cisco routers typically use **flash memory**, with no hard disk drive. Flash memory is rewriteable, permanent storage, which is ideal for storing files that need to be retained when the router loses power. Cisco purposefully uses flash memory rather than hard disk drives in its products because there are no moving parts in flash memory, so there is a smaller chance of failure as compared with disk drives. Some routers have flash memory on the motherboard. Others have flash memory slots that allow easy removal and replacement of the flash card, but with the intent that the card remain in the device most of the time. Also, many devices have USB slots that support USB flash drives.

For each physical memory device in the router, IOS creates a simple IOS file system of type *disk*. In Example 17-1, a Cisco 1108 ISR router has an internal flash memory, with just under 3 GB of memory, which can be referenced as either bootflash:, flash:, or crashinfo:. The bottom of the output confirms that the single USB flash slot holds a flash drive with just under 32 GB, referenced by name usb0:.

Example 17-1 *Cisco IOS File Systems on a Router*

Click here to view code image

```
R2# show file systems
File Systems:
        Size(b)        Free(b)      Type  Flags  Prefixes
              -              -    opaque     rw   system:
              -              -    opaque     rw   tmpsys:
*    2968264704     1416036352      disk     rw   bootflash: flash:
     1634713600     1557024768      disk     ro   webui:
              -              -    opaque     rw   null:
              -              -    opaque     ro   tar:
              -              -   network     rw   tftp:
              -              -    opaque     wo   syslog:
       33554432       33534559     nvram     rw   nvram:
              -              -   network     rw   rcp:
              -              -   network     rw   http:
              -              -   network     rw   ftp:
              -              -   network     rw   scp:
              -              -   network     rw   sftp:
              -              -   network     rw   https:
              -              -    opaque     ro   cns:
    31022530560    31022514176      disk     rw   usb0:
```

Beyond the physical disks, IFS creates other types of disks for use by the operating system, as noted in this list. Example 17-1 also shows examples of each.

- **Opaque:** To represent logical internal file systems for the convenience of internal functions and commands
- **Network:** To represent external file systems found on different types of servers for the convenience of reference in different IOS commands
- **Disk:** For physical disks, either internal or external flash
- **NVRAM:** A special type for NVRAM memory, the default location of the startup-config file

Many IOS commands refer to files in an IFS, but only some commands refer directly to the files by their formal names. The formal names use the prefix as seen in the far right column of Example 17-1. For instance, the command **more flash0:/wotemp/fred** would display the contents of file *fred* in directory */wotemp* in the first flash memory slot in the router. (The **more** command itself displays the contents of a file.) However, many commands use a keyword that indirectly refers to a formal filename, to reduce typing. For example:

- **show running-config** command: Refers to file system:running-config
- **show startup-config** command: Refers to file nvram:startup-config

Upgrading IOS Images

One of the first steps to upgrade a router's IOS to a new version is to obtain the new IOS image and put it in the right location. Typically, Cisco routers have their IOS in one of the local physical file systems, most often in permanent flash. The only requirement is that the IOS be in some reachable file system—even if the file sits on an external server and the device loads the OS over the network. However, the best practice is to store each device's IOS file in flash that will remain with the device permanently.

Figure 17-7 illustrates the process to upgrade an IOS image into flash memory, using the following steps:

Step 1. Obtain the IOS image from Cisco, usually by downloading the IOS image from Cisco.com using HTTP or FTP.

Step 2. Place the IOS image someplace that the router can reach. Locations include TFTP or FTP servers in the network or a USB flash drive that is then inserted into the router.

Step 3. Issue the **copy** command from the router, copying the file into the flash memory that usually remains with the router on a permanent basis.

Key Topic

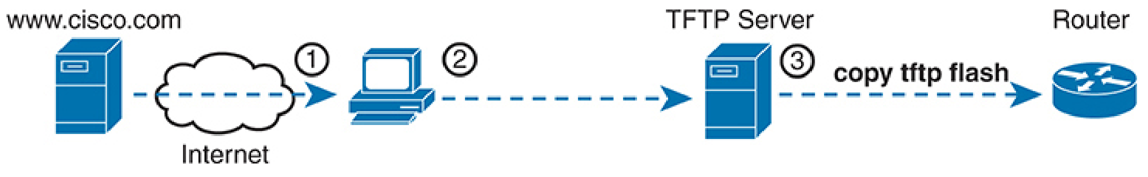


Figure 17-7 *Copying an IOS Image as Part of the Cisco IOS Software Upgrade Process*

Copying a New IOS Image to a Local IOS File System Using TFTP

Example 17-2 provides an example of step 3 from Figure 17-7, copying the IOS image into flash memory. In this case, router R2, a Cisco ISR 1108 router, copies an IOS image from a TFTP server at IP address 2.2.2.1.

Key Topic

Example 17-2 copy tftp flash *Command Copies the IOS Image to Flash Memory*

Click here to view code image

```
R2# copy tftp: flash:
Address or name of remote host []? 2.2.2.1
Source filename []? c1100-universalk9.17.06.03a.SPA.bin
Destination filename [c1100-universalk9.17.06.03a.SPA.bin]?
Accessing tftp://2.2.2.1/c1100-universalk9.17.06.03a.SPA.bin...
Loading c1100-universalk9.17.06.03a.SPA.bin from 2.2.2.1 (via Gig
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 706422748 bytes]

706422748 bytes copied in 187.876 secs (3760047 bytes/sec)
R2#
```

The **copy** command does a simple task—copy a file—but the command also has several small items to check. It needs a few pieces of information from the user, so the command prompts the user for that information by showing the user some text and waiting for the user's input. The bold items in the example show the user's input. The router then has to check to make sure the copy will work. The command works through these kinds of questions:

1. What is the IP address or host name of the TFTP server?
2. What is the name of the file?
3. Ask the server to learn the size of the file, and then check the local router's flash to ask whether enough space is available for this file in flash memory.
4. Does the server actually have a file by that name?
5. Do you want the router to erase any old files in flash?

The router prompts you for answers to some of these questions, as necessary. For each question, you should either type an answer or press **Enter** if the default answer (shown in square brackets at the end of the question) is acceptable. Afterward, the router erases flash memory if directed, copies the file, and then verifies that the checksum for the file shows that no errors occurred in transmission.

Note

Most people use the IOS filenames that Cisco supplies because these names embed information about the IOS image, like the version. Also, if you want to use the same destination filename as the source, avoid the mistake of typing "y" or "yes" to confirm the selection; instead, you would be setting the destination filename to "y" or "yes." Simply press **Enter** to confirm the selection listed in brackets.

Listing the Files in the IOS File System

You can view the contents of the flash file system to see the IOS file that was just copied by using a couple of commands. It helps to think of each IFS as a disk on your favorite desktop operating system, with directories, subdirectories, and files, because those exist in IFS disks as well. For instance, you can use both the **show flash:** and **dir flash:** commands and see files, both showing some evidence of directories.

First, the **show flash:** command displays all files in the flash: file system in one list, regardless of what subdirectory the file resides in. The output lists each file with its directory structure; however, IOS uses both the bootflash: and flash: aliases to represent this disk, so the output lists directories that begin bootflash: rather than flash:. For instance, line 5 in the output in Example 17-3 (the first line with gray highlights) lists

A directory of /bootflash/.installer

A filename of watchlist

Example 17-3 *The* **show flash:** *Command Displays All Files and All Subdirectories*

Click here to view code image

```
R2# show flash:
-#- --length-- ---------date/time---------            path
  1       4096 Aug 01 2022 17:12:23.0000000000 +00:00 /bootflash/
  2       4096 Aug 01 2022 16:12:30.0000000000 +00:00 /bootflash/
  3          5 Aug 01 2022 16:10:40.0000000000 +00:00 /bootflash/
install_global_trans_lock
  4         50 Aug 01 2022 16:10:40.0000000000 +00:00 /bootflash/
last_pkgconf_shasum
  5         11 Aug 01 2022 16:11:15.0000000000 +00:00 /bootflash/
watchlist
! Skipped many lines for brevity...
571  706422748 Jun 27 2022 17:08:17.0000000000 +00:00 /bootflash/
universalk9.17.06.03a.SPA.bin
! Many lines skipped for brevity...
1416802304 bytes available (1400680448 bytes used)
```

The columns of the **show flash:** command provide some key facts. First, IFS numbers the files with a unique number. The # (file number) column lists those numbers. Notice that the **show flash:** command lists the files in file number order. Moving to the right, the length column notes the number of bytes in each file. On the far right, the path column lists the filenames with the entire directory path (beginning with /bootflash: in this case).

Next, look for the line for file 571 near the bottom of Example 17-3, which happens to be the IOS file copied into this disk earlier in Example 17-2. It has a length of just over 700 MB. If you look back to the **copy** command in Example 17-2, you can see the output there that lists the exact size of the file copy, which matches the file size listed by **show flash:**.

The **dir** command lets you display files by directory instead of as a long list of sequential files. To do so, you navigate using the **cd** (change directory) command and discover the current directory with the **pwd** (present working directory) command. IOS begins with a default IFS disk, typically bootflash: (which holds the IOS file). To show the commands in action, Example 17-4 shows these steps:

1. The **pwd** command confirms the default pwd of "bootflash:".
2. Use the **dir** command to lists files/directories in the current directory.
3. Change to a subdirectory of bootflash: (license_evlog).
4. Repeat the **dir** command to list files in the new current directory.
5. Confirm the present working directory again.

Example 17-4 *The* **dir** *Command: Display Files in the Present Working Directory*

Click here to view code image

```
R2# pwd
bootflash:/

R2# dir
Directory of bootflash:/
```

```
88177    drwx            40960   Aug 1 2022 20:34:05 +00:00  trace
64129    drwx             4096   Aug 1 2022 16:12:30 +00:00  .inst
56113    drwx             4096   Aug 1 2022 16:11:48 +00:00  licer
15       -rw-               30   Aug 1 2022 16:11:39 +00:00  throu
12       -rw-           134935   Aug 1 2022 16:11:23 +00:00  memle
11       -rw-             1546   Aug 1 2022 16:11:00 +00:00  mode_
40081    drwx             4096  Jun 27 2022 17:15:02 +00:00  .prst
19       -rw-             1923  Jun 27 2022 17:14:30 +00:00  trust
18       -rw-        706422748  Jun 27 2022 17:08:17 +00:00  c110C
17.06.03a.SPA.bin
! Lines omitted for brevity
2968264704 bytes total (1416704000 bytes free)

R2# cd license_evlog
R2# dir
Directory of bootflash:/license_evlog/


56121    -rw-             1656   Aug 1 2022 16:13:19 +00:00  SAEve
161148.log
56120    -rw-             1927  Jul 29 2022 16:13:37 +00:00  SAEve
161208.log
56119    -rw-             1809   Jul 1 2022 20:28:37 +00:00  SAEve
184140.log
56118    -rw-             1555  Jun 27 2022 17:22:22 +00:00  SAEve
171513.log

2968264704 bytes total (1416802304 bytes free)

R2# pwd
bootflash:/license_evlog/
```

Finally, take a moment to compare the usage memory usage statistics from the **show flash:** (Example 17-3) and **dir** (Example 17-4) commands. Both commands end with one line showing memory use statistics. Both list the number of free (unused) bytes; however, the **dir** command lists total bytes (used + unused), while the **show flash:** command lists total bytes used. Make sure you understand the difference. Then use these statistics to check if enough memory exists to copy in new IOS images, and if not, you should expect to remove some old files to make room.

Verifying IOS Code Integrity with MD5 or SHA512

You download the IOS from Cisco, copy it to your router, and run it. Is it really the code from Cisco? Or did some nefarious attacker somehow get you to download a fake IOS that has a virus?

Cisco provides a means to check the **code integrity** of the IOS file to prevent this type of problem by creating a hash value based on the IOS image. First, when Cisco builds a new IOS image, it calculates and publishes both an MD5 and SHA512 hash value for that specific IOS file. To create the MD5 hash, Cisco uses as input the IOS file itself and runs the MD5 math algorithm against that file. To create the SHA512 hash, Cisco uses the same process, but with the SHA512 algorithm.

To verify that the IOS file on your router or switch has not been changed, use the **verify** command to re-create the same math that Cisco used to create the hash values. To do so, follow the process shown in Figure 17-8. As shown on the right side of the figure, first, find the hash(es) created by Cisco by finding your IOS file in the downloads area of cisco.com. Then, you run that same MD5 or SHA512 math on your router against the IOS file on the router, using the IOS **verify** command. That command will generate the same hash as listed at the Cisco site: if both hashes are equal, the file has not changed.

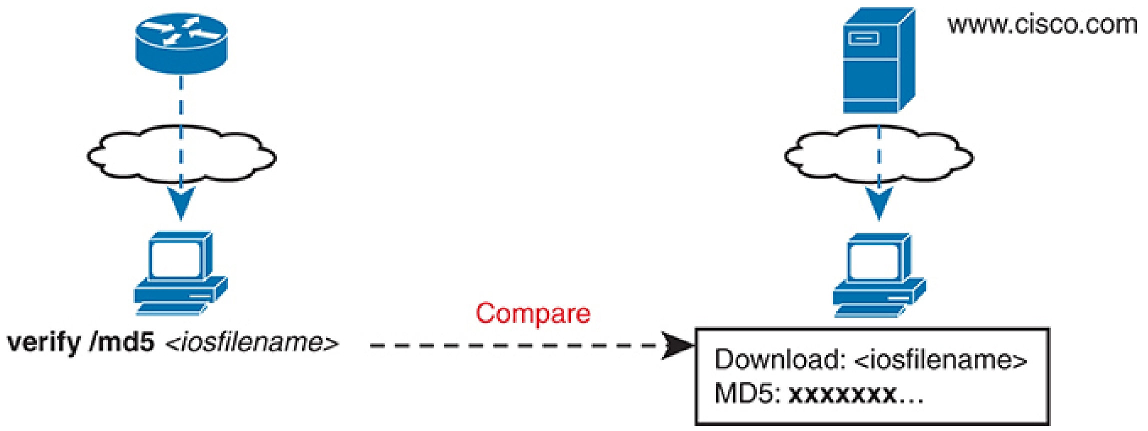


Figure 17-8 *MD5 Verification of IOS Images—Concepts*

The **verify /sha512** command generates the SHA512 hash on your router, as shown in Example 17-5. Note that you can include the hash value computed by Cisco as the last parameter or omit it. If you include it (as in the example), IOS will compare the locally computed value to the value you pasted into the command and tell you if they match. If you omit the value from the command, the **verify** command lists the locally computed SHA512 hash, but you have to do the picky character-by-character check of the values yourself.

Example 17-5 *Verifying Flash Memory Contents with the* **verify** *Command*

Click here to view code image

```
R2# verify /sha512 flash0:c1100-universalk9.17.06.03a.SPA.bin 164
255f2271ccf6fc633d4abb595d46b13dd280811aca3313a080f6282efe2365076
00b8ac558785b4090c869f8548
...................................................................
...................................................................
(many lines of periods skipped for brevity)...
Done!
Verified (bootflash:/c1100-universalk9.17.06.03a.SPA.bin) = 16419
2271ccf6fc633d4abb595d46b13dd280811aca3313a080f6282efe2365076985f
00b8ac558785b4090c869f8548
```

Copying Images with FTP

The networking world has many options for file transfer, several of which IOS supports for the transfer of files into and out of the IOS file systems that reside on the router. TFTP and FTP have been supported for the longest time, with more recent support added for protocols like Secure Copy Protocol (SCP), which uses the SSH File Transfer Protocol (SFTP). Table 17-2 lists some of the names of file transfer protocols that you might come across when working with routers.

Table 17-2 Common Methods to Copy Files Outside a Router

Method	Method (Full Name)	Encrypted?
TFTP	Trivial File Transfer Protocol	No
FTP	File Transfer Protocol	No
SCP	Secure Copy Protocol	Yes

To copy files with FTP, you follow the same kind of process you use with TFTP (see Example 17-6). You can follow the interactive prompts after using an EXEC command like **copy ftp flash:**. However, the **copy** command allows you to use a URI for the source and/or destination, which lets you put most or all of the information in the command line itself. Each URI refers to the formal name of a file in the IFS.

Key Topic

Example 17-6 *Installing a New IOS with FTP*

Click here to view code image

```
R1# copy ftp://wendell:odom@192.168.1.170/c1100-universalk9.17.06
Destination filename [c1100-universalk9.17.06.03a.SPA.bin]?
Accessing ftp://192.168.1.170/c1100-universalk9.17.06.03a.SPA.bir
Loading c1100-universalk9.17.06.03a.SPA.bin
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 706422748/4096 bytes]

706422748 bytes copied in 119.604 secs (5906544 bytes/sec)
```

First, take a close look at the long URI in the command that begins with "ftp." The "ftp" part identifies the protocol, of course. After the //, the text references the username (wendell) and password (odom), as well as the FTP server's IP address. After the single / comes the filename on the server.

Although the command is long, it has only two parameters, with the long first parameter and the short keyword **flash:** as the second parameter. The **copy** command lists the source location as the first parameter and the destination as the second. The destination in this case, **flash:**, refers to the internal flash file system, but it does not mention the filename. As a result, IOS prompts the user for a specific destination filename, with a default (in brackets) to keep the source filename. In this case, the user just pressed Enter to accept the default. To avoid being prompted at all, the command could have listed **flash:c1100-universalk9.17.06.03a.SPA.bin** as that second parameter, fully defining the destination file.

Finally, with another twist, you can configure the FTP username and password on the router so that you do not have to include them in the **copy** command. For instance, the global configuration commands **ip ftp username wendell** and **ip ftp password odom** would have configured those values. Then the **copy** command would have begun with **copy ftp://192.168.1.170/...**, omitting the username:password in the command, without needing to then prompt the user for the username and password.

The FTP and TFTP Protocols

The IOS **copy** command, when using the **tftp** or **ftp** keyword, makes the command act as a client. The client connects to a TFTP or FTP server and then attempts to transfer the file. In the examples from the IOS, that **copy** command copied the file from the server into the client device (a router).

The rest of this section takes a closer look at both FTP and TFTP as protocols and tools.

FTP Protocol Basics

FTP has long been a core Internet protocol, serving as the primary file transfer protocol for several decades. RFC 959, which standardizes FTP, dates back to 1985. FTP uses TCP as its transport protocol, relying on TCP to provide an error-free in-order delivery of data so that the FTP application knows that each file transfer creates an exact copy of the file with no omissions. FTP uses well-known TCP port 21, and in some cases, also uses well-known port 20.

As for normal operation, FTP uses a client/server model for file transfer, as shown in the example in Figure 17-9. The figure shows the major steps but not every message. For instance, step 1 shows host A creating a TCP connection to the server (which takes the usual three TCP messages). Step 2 represents the exchange that allows the server to authenticate the client. Step 3 shows the idea that, once authenticated, the client and server can send FTP commands over the connection to tell the other device what to do.

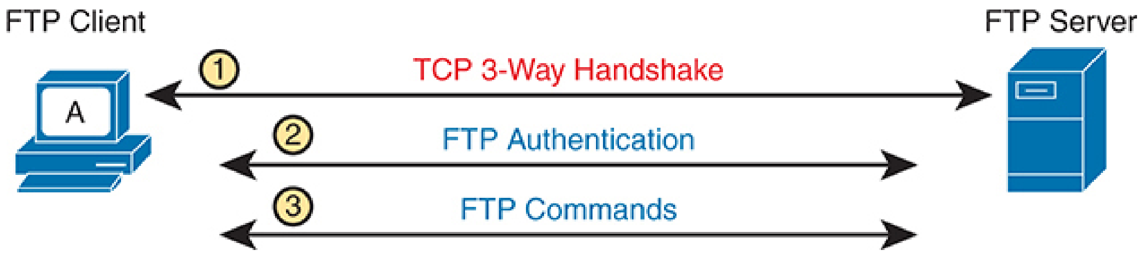


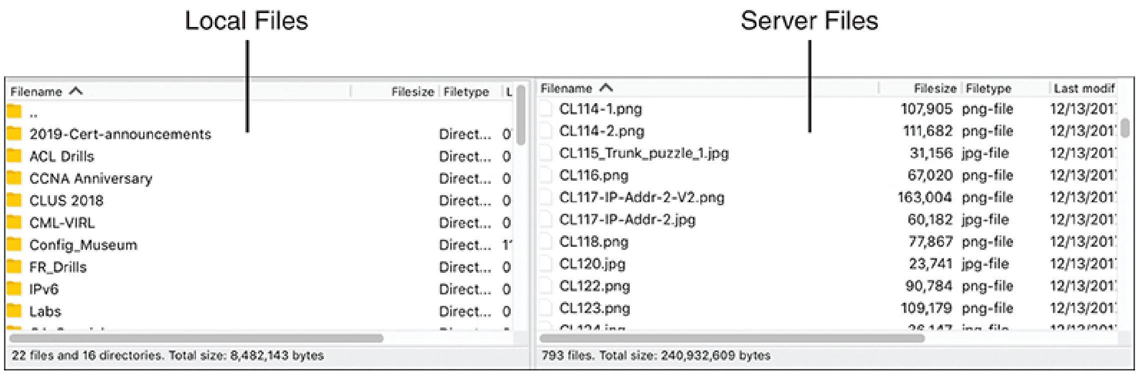
Figure 17-9 *Major Concepts with FTP Clients and Servers*

The commands that flow over this initial TCP connection—called the **FTP control connection**—define the kinds of functions supported by FTP. Those commands allow the client to navigate around the directory structures of the server, list files, and then transfer files from the server (FTP GET) or to the server (FTP PUT). A summary of some of the FTP actions is as follows:

Key Topic

- **Navigate directories:** List the current directory, change the current directory to a new directory, go back to the home directory, all on both the server and client side of the connection.
- **Add/remove directories:** Create new directories and remove existing directories on both the client and server.
- **List files:** List files on both the client and server.
- **File transfer:** Get (client gets a copy of the file from the server), Put (client takes a file that exists on the client and puts a copy on the FTP server).

While many OSs support command-line FTP clients, which require you to learn the various FTP commands and use those from the command line, most users instead use an FTP client app that issues the FTP commands behind the scenes. Clients typically display files on the local system as well as the server with a user interface that has a similar appearance as a file browser on a desktop OS (for instance, Windows Explorer, macOS Finder). Figure 17-10 shows a sample user interface from the FileZilla FTP client (Filezilla-project.org).



Filename	Filesize	Filetype	L
..			
2019-Cert-announcements		Direct...	0
ACL Drills		Direct...	0
CCNA Anniversary		Direct...	0
CLUS 2018		Direct...	0
CML-VIRL		Direct...	0
Config_Museum		Direct...	1
FR_Drills		Direct...	0
IPv6		Direct...	0
Labs		Direct...	0

22 files and 16 directories. Total size: 8,482,143 bytes

Filename	Filesize	Filetype	Last modif
CL114-1.png	107,905	png-file	12/13/201
CL114-2.png	111,682	png-file	12/13/201
CL115_Trunk_puzzle_1.jpg	31,156	jpg-file	12/13/201
CL116.png	67,020	png-file	12/13/201
CL117-IP-Addr-2-V2.png	163,004	png-file	12/13/201
CL117-IP-Addr-2.jpg	60,182	jpg-file	12/13/201
CL118.png	77,867	png-file	12/13/201
CL120.jpg	23,741	jpg-file	12/13/201
CL122.png	90,784	png-file	12/13/201
CL123.png	109,179	png-file	12/13/201

793 files. Total size: 240,932,609 bytes

Figure 17-10 *FTP Client Example with FileZilla*

The client application in Figure 17-10 lists the client computer's local file system on the left and the FTP server's file system on the right. The user can click on the right to change directories, much like using any app that browses a file system, with FTP performing the commands behind the scenes. The user can also drag and drop files from the left to the right to put a file on the server, or vice versa to get a file from the server.

The FTP server can be a server application installed and managed by others, or you can install or enable an FTP server for your own use. For instance, a network engineer might install an FTP server application on a laptop for use in upgrading IOS files, while the IT staff may keep an FTP server available 24/7 for all employees of the company to use. A simple Internet search can show a variety of FTP server applications that run on the common desktop OSs. Additionally, both Windows and macOS come with an FTP or FTPS (FTP Secure) server option built into the OS; all you have to do is enable it. (The Linux distributions all have FTP servers available via simple downloads.) (The aforementioned FileZilla also has a free open-source FTP server option, which can be an easy way to get started.)

Once it is installed, the server can be configured with a variety of settings. For instance, the server needs to specify which users can access the server, so it can use the same login credentials allowed for the host where it resides or specify other credentials. It can specify the directories that each user can access, and whether the user has read-only or read-write access.

FTP Active and Passive Modes

FTP can operate in either active or passive mode. The choice of mode may impact whether the FTP client can or cannot connect to the server and perform normal functions. The user at the FTP client can choose which mode to use, so this section works through the underlying details to explain why FTP passive mode may be the more likely option to work.

First, note that FTP uses two types of TCP connections:

Key Topic

- **Control Connection:** Used to exchange FTP commands
- **Data Connection:** Used for sending and receiving data, both for file transfers and for output to display to a user

Given the two roles, when a client connects to an FTP server, the client first creates the FTP control connection as shown in Figure 17-11. The server listens for new control connections on its well-known port 21; the client allocates any new dynamic port (49222 in this case) and creates a TCP connection to the server.

Key Topic

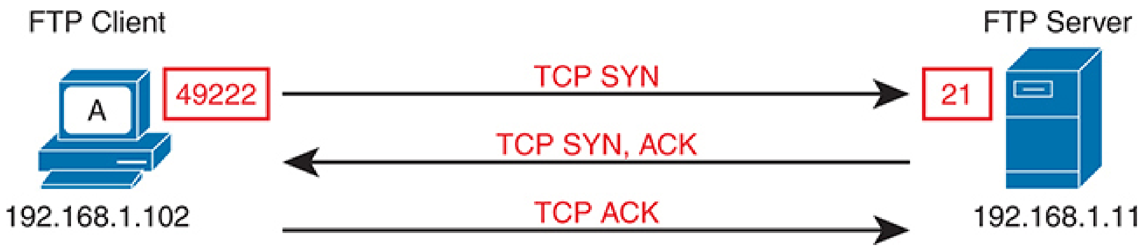


Figure 17-11 *FTP Client Creates an FTP Control Connection*

After creating the TCP connection, the user authenticates to the FTP server and takes some actions. Some of those actions require only the control connection, but eventually the user will take an action (like getting a file) that requires a data connection. When that happens, to create the **FTP data connection**, the client will either use active mode or passive mode, as shown in the next two examples.

Figure 17-12 shows an example of what happens in active mode. Following the steps in the figure:

1. The FTP client allocates a currently unused dynamic port and starts listening for new connections on that port.
2. The client identifies that port (and its IP address) to the FTP server by sending an FTP **PORT** command to the server over the control connection.
3. The server, because it also operates in active mode, expects the **PORT** command; the server reacts and initiates the FTP data connection to the client's address (192.168.1.102) and port (49333).

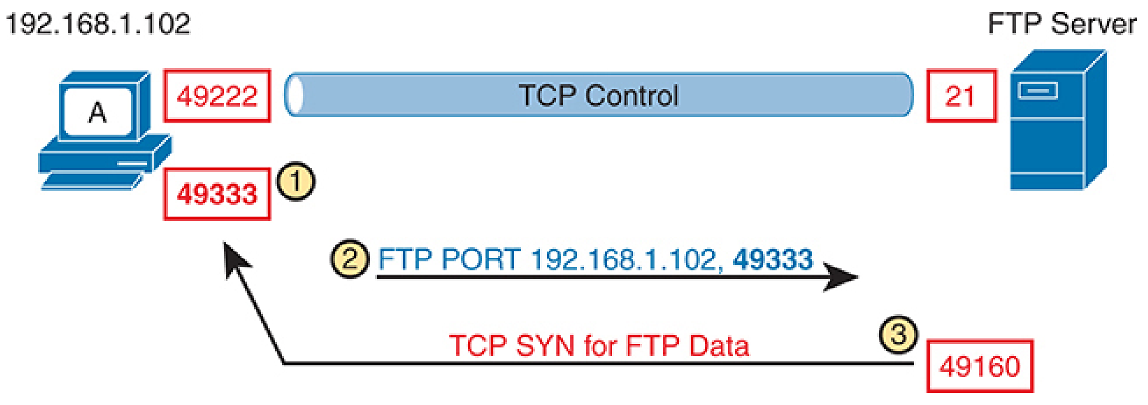


Figure 17-12 *FTP Active Mode Process to Create the Data Connection*

Note that in active mode, the server initiates the TCP connection for data. That works well with both the FTP client and server sitting inside the same enterprise network. When within the same network, typically no NAT function and no firewall sit between the two.

However, if the FTP client sits in an enterprise network, and the FTP server resides somewhere in the Internet, an active mode connection typically fails. Most firewalls do not allow Internet-based hosts to initiate TCP connections to hosts inside the enterprise without a specific firewall rule allowing connections to a known port, and in this case, the FTP client allocates any available port number. For instance, in Figure 17-12, the TCP connection (step 3) would be discarded by a firewall.

Note

When using Active mode, the server may use its well-known data port 20 or a dynamic port. (Figure 17-12 shows the use of dynamic port 49160.)

Passive mode helps solve the firewall restrictions by having the FTP client initiate the FTP data connection to the server. However, passive mode does not simply cause the FTP client to connect to a well-known port on the server; it requires more exchanges of port numbers to use between the server and client, as shown in Figure 17-13, with these steps:

1. Via messages sent over the control connection, the FTP client changes to use FTP passive mode, notifying the server using the FTP **PASV** command.
2. The server chooses a port to listen on for the upcoming new TCP connection, in this case TCP port 49444.
3. Again using the control connection, the FTP server notifies the FTP client of its IP address and chosen port with the FTP **PORT** command.
4. The FTP client opens the TCP data connection to the IP address and port learned at the previous step.

Key Topic

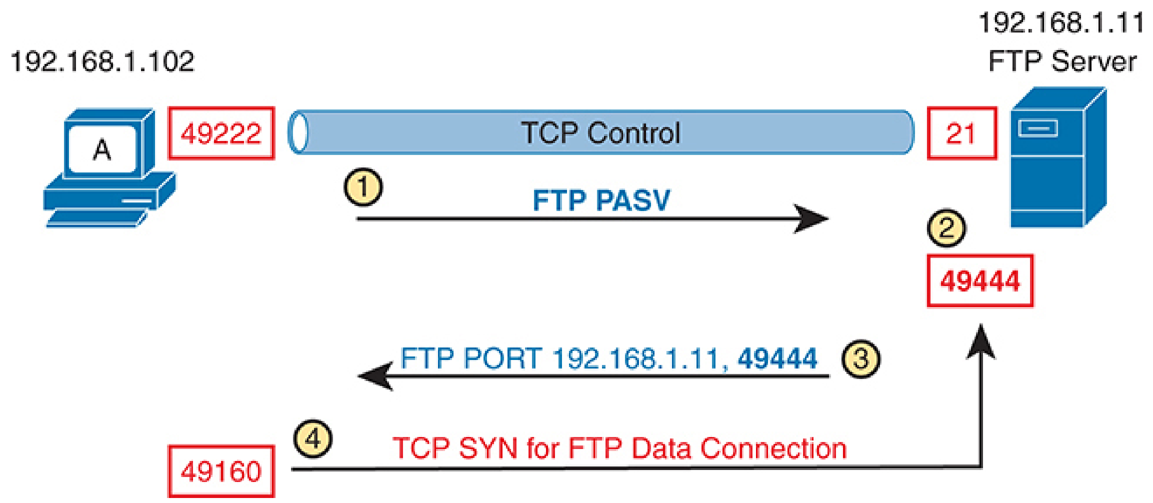


Figure 17-13 *FTP Passive Mode Process to Create the Data Connection*

TFTP Protocol Basics

FTP has a role as a general file transfer tool for any user, with a good number of FTP client application options available. TFTP plays a much smaller role as a tool for the average user, but it does play a more useful role for IT support staff.

For the basics, Trivial File Transfer Protocol uses UDP well-known port 69. Because it uses UDP, TFTP adds a feature to check each file for transmission errors by using a checksum process on each file after the transfer completes.

Key Topic

The word *trivial* in the name refers to its relatively small number of features, meant to be an advantage by making the tool lightweight. For instance, TFTP supports far fewer commands than FTP (fewer functions), meaning that the code requires less space to install, which can be useful for devices with limited memory. TFTP can Get and Put files, but it includes no commands to change directories, create/remove directories, or even to list files on the server. TFTP does not support even simple clear-text authentication. In effect, if a TFTP server is running, it should accept requests from any TFTP client.

Ideally, TFTP has its best use as a temporary tool for quick file transfers in a controlled environment, particularly when the data itself does not have to be secure. For instance, imagine this scenario:

1. A network engineer keeps all router and switch IOS images in a folder.
2. The engineer enables a TFTP server on a laptop as needed; otherwise, the TFTP server remains disabled.
3. The engineer connects the laptop to a LAN and enables the TFTP server long enough to transfer IOS images into or out of a few devices.
4. If the engineer forgets to disable TFTP, the only risk is that someone may copy an IOS image—an image that is already available from Cisco.com to any customer.

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 for more details. Table 17-

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 17-3 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 All the Key Topics

Key Topic

Table 17-4 Key Topics for Chapter 17

Key Topic Element	Description	Page Number
Figure 17-2	The SNMP Get Request and Get Response message flow	371
Figure 17-3	SNMP notification with SNMP Trap messages	372
Figure 17-6	The use of SNMP RO and RW communities with SNMP Get and Set	375
List	SNMP security benefits	375
Figure 17-7	Process of upgrading IOS using TFTP	378
Example 17-2	Example of using TFTP to load new IOS	378
Example 17-6	Example of using FTP to load new IOS	383

Key Topic Element	Description	Page Number
List	FTP functions	384
List	FTP data and control connections	386
Figure 17-11	FTP Control connection establishment	386
Figure 17-13	FTP data connection establishment in passive mode	387
Paragraph	Description of limited functions of TFTP	388

Key Terms You Should Know

code integrity
flash memory
FTP
FTP control connection
FTP data connection
IOS file system
IOS image
Management Information Base (MIB)
Network Management System (NMS)
read-only community
read-write community
Simple Network Management Protocol (SNMP)
SNMP agent
SNMP community
SNMP Get
SNMP Inform
SNMP manager
SNMP Set
SNMP Trap
SNMPv2c
SNMPv3

TFTP

Command References

Tables 17-5 and 17-6 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 17-5 Chapter 17 Configuration Command Reference

Command	Description
boot system flash [*flash-fs*:] [*filename*]	Global command that identifies the location of an IOS image in flash memory
boot system {**tftp** | **ftp**} *filename* [*ip-address*]	Global command that identifies an external server, protocol, and filename to use to load an IOS from an external server
ip ftp username *name*	Global command to define the username used when referencing the **ftp:** IOS file system but not supplying a username
ip ftp password *pass*	Global command to define the password used when referencing the **ftp:** IOS file system but not supplying a password

Table 17-6 Chapter 17 EXEC Command Reference

Command	Description
copy *from-location to-location*	Enable mode EXEC command that copies files from one file location to another. Locations include the startup-config and running-config files, files on TFTP and RPC servers, and flash memory.

show flash:	Lists the names and size of the files in internal flash memory, and notes the amount of flash memory consumed and available.
dir *filesystem:/* **dir** *filesystem:directory*	Lists the files in the referenced file system or file system directory.
verify {/md5 | /sha512} *filesystem:name [hash-value]*	Performs an MD5 or SHA512 hash of the referenced file and displays the results. If listed, the command compares the hash in the command with the results of performing MD5 or SHA512 hash on the local file.
cd *directory-name*	If the directory exists, this command changes the present working directory (pwd) variable to refer to the listed directory.
cd ..	Changes the present working directory setting to one directory above the current directory.
pwd	Displays the present working directory, a variable used by commands like **dir** to identify a file system directory in which to check for files needed for a command.

Part IV Review

Keep track of your part review progress with the checklist shown in Table P4-1. Details on each task follow the table.

Table P4-1 Part 4 Review Checklist

Activity	1st Date Completed	2nd Date Completed
Repeat All DIKTA Questions		
Answer Part Review Questions		
Review Key Topics		
Do Labs		
Watch Video		
Use Per-Chapter Interactive Review		

Repeat All DIKTA Questions

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

Answer Part Review Questions

For this task, use PTP to answer the Part Review questions for this part of the book.

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 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 (https://www.certskills.com) includes a series of configuration-focused labs that you can do on paper, each in 10–15 minutes. Review and performs labs for this part of the book by using the menus to navigate to the per-chapter content and then finding all config labs related to that chapter. (You can see more detailed instructions at https://www.certskills.com/config-labs).

Other: If using other lab tools, here are a few suggestions: All the exam topics in Part IV that include the word configure exist in Chapters 13 and 14, so focus on those chapters. Those chapters touch on CDP/LLDP, NTP, syslog, and NAT/PAT.

Watch Video

The companion website includes a variety of common mistake and Q&A videos organized by part and chapter. Use these videos to challenge your thinking, dig deeper, review topics, and better prepare for the exam. Make sure to bookmark a link to the companion website and use the videos for review whenever you have a few extra minutes.

Use Per-Chapter Interactive Review

Using the companion website, browse through the interactive review elements, like memory tables and key term flashcards, to review the content from each chapter.