Chapter 1
Configuring the System

1

This chapter focuses on describing how you can configure basic system-wide, interface, and line capabilities. The following tasks are addressed:

- Customizing Cisco protocol translator options with the **configure** command
- Establishing passwords and maintaining system security using Cisco's Terminal Access Controller Access System (TACACS)
- Configuring protocol translators to support the Simple Network Management Protocol (SNMP)
- Redirecting system messages
- Configuring console and terminal lines
- Configuring the network interfaces

Note: Protocol-specific configuration features are detailed in the protocol-specific chapters of this manual. In addition, simple configuration processes using the **setup** command facility are provided in the "First-Time Startup" chapter. The "Using the Protocol Translator" chapter provides an overview of system operation and user-level commands available with the protocol translator.

Preparing to Configure Your Protocol Translator

Use the privileged EXEC command **configure** to begin configuration of the protocol translator.

Begin by entering the privileged level of the EXEC. This is done by entering the **enable** command at the EXEC prompt:

```
CPT>enable
```

The EXEC then prompts you for the privileged-level password:

```
Password:
```

Type in the password. For security purposes, the password will not be displayed. (Also note that the password is case-sensitive.) When you enter the correct password, the system displays the privileged mode system prompt:

```
CPT#
```

To begin configuration mode, enter the **configure** command at the privileged mode prompt:

```
CPT#configure
```

When you enter this command, the EXEC prompts you for the source of the configuration subcommands.

```
Configuring from terminal, memory, or network [terminal]?
```

The default is to type in commands from the terminal console. Pressing the Return key begins this configuration method. Each configuration technique—terminal, memory, and network—is described in more detail later in this chapter.

The EXEC provides you with a simple editor for entering the configuration commands, and explains the editing functions:

```
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U;end with CTRL/Z
```

Table 1-1 lists the edit key functions and their meanings.

Table 1-1 Configuration Edit Keys

Key	Meaning
Delete or Backspace	Erases one character.
Ctrl-W	Erases a word.
Ctrl-U	Erases a line.
Ctrl-R	Redisplays a line.
Return	Executes single-line commands.
Ctrl-Z	Ends configuration mode and returns to the EXEC.

Note: Use the **disable** EXEC command to return to the user command mode.

Each configuration technique (terminal, memory, and network) is described in more detail later in this chapter.

Configuration Overview

To reduce user input, the protocol translator software provides default settings that set up standard configurations. The **setup** command facility described in the "First-Time Startup" chapter establishes baseline protocol translation capabilities; however, you will need to

provide specific information about your environment to establish operation of your protocol translator.

As an example, the following list summarizes the prerequisites for basic operation when translating between LAT, Telnet, and X.25:

- Configure the Ethernet interface and enter an Internet address. Internet addresses are assigned by the Defense Data Network Information Center (NIC). You will also need to enter the subnet mask you will use and assign a default gateway address if you are not using Proxy Address Resolution Protocol (ARP).
- Configure the X.25 interface and enter the X.25 address. The X.25 address is typically provided by the X.25 service provider.
- Adjust the X.25 Level 2 and Level 3 parameter settings for your application.
- Enable LAT on serial lines, if necessary, and specify any required inbound session support.
- Establish protocol translation for both incoming and outgoing connections.

Additionally, the protocol translator provides commands for configuring the terminal lines and establishing network management, message logging, and line security. The subcommands for each task are listed in alphabetical order within the separate sections describing each task.

You use three privileged commands to configure your protocol translator: the EXEC command **configure,** the **interface** configuration command, and the **line** configuration command.

- The EXEC command **configure** starts the initial system configuration process by putting you in configuration mode.

  After entering the **configure** command at the system prompt, the EXEC prompts you for the source of the configuration commands. You can type in the configuration commands from the console terminal, read them in from non-volatile memory, or read them in from a file on a remote host. Each technique is described in greater detail in the "Configuration Methods" section later in this chapter.
- The global configuration subcommands are used to establish such things as network services, a default configuration file name, and the TACACS server. Most importantly, the global **translate** command establishes protocol translation.
- The **interface** configuration subcommands allow you to specify the parameters that configure your network interface. The network interfaces for the CPT are Ethernet and serial. The IGS can have two Ethernet interfaces, or one Ethernet and one serial interface.
- The **line** configuration subcommands allow you to specify the parameters that configure your terminal lines.

Entering the Configuration Commands

The configuration subcommands are categorized by these functions:

- Global configuration commands—Define system-wide parameters.
- Interface subcommands—Define the characteristics of an interface (a serial or Ethernet interface, for example) and must be preceded by an **interface** command.
- Line subcommands—Define the characteristics of a serial terminal line and must be preceded by a **line** command.

The descriptions of the commands include the command type and give examples of their use.

As with EXEC commands, you can type configuration subcommands in uppercase letters, lowercase letters, or both. You may also shorten all commands and other keywords to unique abbreviations. You may add comments by preceding the line with an exclamation point (!). Comments do not affect command processing.

If you make a typing mistake, use the Delete or Backspace key to erase a character, Ctrl-W to erase a word, and Ctrl-U to erase a line. To redisplay a line, use Ctrl-R. See Table 2-1for a list of valid commands.

The network server executes single-line commands when you press the Return key. The network server does not display confirmation messages as it executes the commands. If the network server encounters a problem, it displays an error message on the console terminal. When you type Ctrl-Z, the network server exits the configuration mode.

In most cases, you can negate a configuration subcommand or restore a default by typing **no** before the subcommand keyword. You can usually omit the arguments of the subcommand when you negate it with **no.** The command descriptions note any exceptions to these rules.

Examples of Configuration Files

Following are some examples of configuration files to illustrate how to enter the configuration commands.

Global Configuration Commands

Use global configuration commands to enable functions that affect the entire system rather than a particular line or interface, and can appear any place within the configuration file. An example of this is the global configuration command to define the host name, or the name of the protocol translator:

```
hostname proto-1
```

Interface Subcommands

Interface subcommands modify the operation of an interface such as an Ethernet, FDDI, or serial port. Interface subcommands always follow an **interface** command which defines the interface type.

The following example illustrates how to enable LAT and DEC MOP on interface *Ethernet 0*:

```
interface ethernet 0
mop enabled
lat enabled
```

Note: The EXEC accepts commands in uppercase and lowercase letters. Exclamation points (!) are not parsed and serve as comment lines and delimiters between configuration commands.

If you forget to enter the **interface** command, the system displays the message "must specify a network interface."

Line Subcommands

Line subcommands modify the operation of a serial terminal line. Line subcommands always follow a **line** command which defines the line number. If you forget to enter the **line** command, the system displays the message "must specify a line or range of lines."

The following example illustrates how to set the password on line 5:

```
line 5
password secretword
```

Type Ctrl-Z to end your configuration sessions, and to use the **disable** command to leave privileged level mode.

Configuration Methods

The EXEC allows you to enter the configuration subcommands from the console terminal, read the subcommands in from non-volatile memory, or read them in from a file on a remote host. The EXEC offers three commands that place the configuration information in these places for you.

Use the EXEC command **write terminal** to display current configuration information on your terminal screen.

Use the EXEC command **write memory** to copy current configuration information to non-volatile memory. This command stores all non-default configuration information as

configuration commands in text format. This command also records a checksum for the information to protect against data corruption.

Use the EXEC command **write network** to copy the current configuration information to a server host on the network.

Use the EXEC command **show configuration** to display information stored in non-volatile memory. You can use this command and the **write terminal** command to find differences between the current configuration and that stored in non-volatile memory.

Use the EXEC command **write erase** to clear the contents of non-volatile memory.

Once you have specified an interface address and written information to non-volatile memory, the software, by default, looks for configuration information on a network host. Procedures later in this section explain how to set up configuration files on a remote host, and how to prevent this process, if you prefer not to store configuration information this way.

Note: The **write** commands create their output by examining the state of the system currently running. The output produced by the **write** commands is generated by the software and will not necessarily match the text the user entered to create the current configuration.

Configuring From the Console

To specify configuration commands and subcommands from the console terminal, enter the EXEC command:

configure

The protocol translator responds with a prompt asking you to specify the terminal, a file, or non-volatile memory as the source of configuration subcommands. To begin configuration, press the Return key or type **terminal** at the prompt displayed by the **configure** command to start command collection.

During command execution, the protocol translator accepts one configuration command per line. You can enter as many configuration subcommands as you want. Type Ctrl-Z when you finish entering configuration commands. This returns you to the EXEC where you can test your configurations or write the configuration commands to memory.

After making changes, write the configuration information into non-volatile memory or to a configuration file stored on a remote host. This will make checking, adding information to, and booting the configuration file an easier task. The procedures for writing information to non-volatile memory are described next.

Automatic Configuration Using Non-volatile Memory

After you enter the desired configuration information at the console terminal, use the privileged EXEC command:

write memory

This EXEC command makes a copy of the configuration information in the non-volatile memory. The memory stores the current configuration information in text format as configuration commands, recording only nondefault settings. The memory is checksummed to guard against corrupted data.

As part of its startup sequence, the protocol translator startup software always checks for configuration information in the non-volatile memory. Once the non-volatile memory holds valid configuration commands, the protocol translator executes the commands automatically at startup. If the protocol translator detects a problem with the non-volatile memory or the configuration information it contains, the protocol translator may enter the setup mode, prompting for configuration information. Problems can include a bad checksum for the information in the non-volatile memory and the absence of critical information.

Even if the non-volatile memory provides configuration information, the protocol translator can be made to attempt to load additional configuration information from the network server. You may want to keep an up-to-date version of configuration information on another host, where you can change it as necessary, and use the non-volatile memory as a bootstrap or backup mechanism. Instruct the protocol translator to make this attempt by placing the following command in the non-volatile memory:

service config

The default is an implied **no service config**. This command limits the source of automatic configuration information to the non-volatile memory.

To display the configuration information stored in the non-volatile memory, use the privileged EXEC command:

show configuration

To clear the contents of the non-volatile memory, use the privileged EXEC command:

write erase

To re-execute the configuration commands stored in non-volatile memory, enter the command at the configure mode prompt:

configure memory

Using the **configure memory** command is like performing a soft reset of the system.

Automatic Configuration Using Remote Hosts

To store configuration information on a remote host, use the privileged EXEC command:

write network

This command will prompt you for the destination host's IP address and a file name.

To retrieve and/or add to the configuration information stored on a host file, enter **configure** and then enter the **network** command at the configure mode prompt:

network

Note: For more information about loading network or host configuration files over the network and specifying boot files, refer to the "Setting Configuration File Specifications" section later in this chapter and to the "Loading Software over the Network" section in the "Managing the System" chapter.

The protocol translator may be configured to automatically load additional configuration information from a network host. You may want to keep an up-to-date version of configuration information on another host, where you can change it as necessary, and use the non-volatile memory as a bootstrap or backup mechanism. You can instruct the protocol translator to load configuration information over the network by entering the **service config** subcommand and then writing the information to non-volatile memory using the **write memory** command.

Note: Loading configuration file information from a remote host is the default, if non-volatile memory is not installed.

After loading configuration information from the non-volatile memory, the protocol translator will attempt to load two configuration files from remote hosts. The first is the network configuration file, which contains commands that apply to all protocol translators on a network. The second is the host configuration file, which contains commands that apply to one protocol translator in particular.

The default name of the network configuration file is "network-confg." The default name for the host configuration file is taken from the host name. The host name can be specified by the **hostname** configuration subcommand or can be derived from the Domain Name System (DNS); for more information, refer to the section "Setting the Host Name" further ahead in this chapter. To form the host configuration file name, the protocol translator converts the host name to lower case, stripped of any DNS information, and appends "-confg." If no host name information is available, the default host configuration file name is "cpt-confg." Other names for these configuration files can be set using the **boot** command, which is described in the "Setting Configuration File Specifications" section further ahead in this chapter.

The protocol translator uses TFTP to load and save configuration files. By default, the protocol translator uses an Internet address of all ones (255.255.255.255) to broadcast TFTP Read Request messages. However, many hosts use an old style of broadcast address consisting of all zeros. You can change operation to accommodate hosts using the old style of broadcast address (0.0.0.0.) using the **ip broadcast-address** interface subcommand.

Note: TFTP is the Trivial File Transfer Protocol defined in RFC 783. The details of setting up a TFTP server process and installing the configuration files on the server host vary from one operating system to another; refer to the documentation for your host computer if you need more information about TFTP support.

If the protocol translator fails to load a configuration file during startup, it tries again every ten minutes (default setting) until a host provides the requested files. With each failed attempt, the protocol translator displays a message at the console terminal.

Example:

If the protocol translator is unable to load the file named "network-confg," it displays the message:

```
Booting network-confg ... [timed out]
```

To end these file load attempts, enter the following configuration command at the console terminal and save it in the non-volatile memory:

no service config

This command prevents the protocol translator from trying to access nonexistent TFTP servers when it is booted.

Note: Be aware that the system treats network and host configuration files differently when loading new parameters. When a host configuration file is loaded, all terminal line parameters are cleared before setting any new parameters. When a network configuration file is loaded, no old parameters are cleared. This means that terminal line parameters set by the network configuration file, which are generally loaded first, will be reset by the host configuration file, which is generally loaded second.

Automatic Configuration Using Network Servers

Automatic configuration via network servers requires the protocol translator first to determine its interface address from network servers, then to load configuration files.

Determining the Protocol Translator's Address

The protocol translator must identify the Internet address of its interface before it can complete its startup sequence. The protocol translator first tries to find the required address in the non-volatile memory. If the attempt succeeds, the protocol translator continues with the next step in the startup sequence, loading operating software or configuration information. If the attempt fails, the protocol translator tries to get address information from other network servers.

To get address information from other network servers, the protocol translator sends a broadcast request to all network servers. If a network server responds and supplies the required address information, the protocol translator proceeds to load operating software or configuration information. If no network server responds, the protocol translator displays an address request prompt on the console terminal and continues periodic network queries. The protocol translator waits indefinitely for input from the console terminal or for a response from the network.

The protocol translator uses different techniques for determining the address, depending on the type of network interface. For an Ethernet interface, the protocol translator uses Reverse Address Resolution Protocol (RARP) and, if RARP fails, Boot Protocol (BootP). Both protocols send a broadcast message to all available network servers. The message requests the 32-bit Internet address corresponding to the 48-bit hardware address of the Ethernet interface.

Many UNIX systems support RARP and BootP, which are defined in RFC 903 and RFC 951, respectively.

Loading the Configuration File

After loading configuration information from the non-volatile memory (if any), the protocol translator attempts to load two configuration files from other network servers. The network configuration file contains commands that apply to all Cisco Systems protocol translators and network servers on a network host. The host configuration file contains commands that apply to one protocol translator in particular. If non-volatile memory is included with the system, it must include the **service config** command in order to cause the system to read configuration files from the network.

The first file the protocol translator tries to load is the network configuration file "network-confg." Typically, the network configuration file contains commands to specify a default gateway, a default domain name, a list of name servers, and a set of host-name-to-address bindings. You can specify a new network configuration file name by putting the **boot network** command into the non-volatile memory.

After loading the network configuration file, the protocol translator establishes its name. The protocol translator first looks for name-to-address information provided by an **ip host** command in the network configuration file. If this attempt fails, the protocol translator broadcasts a Domain Name System-based address-to-name query. If the protocol translator receives no response, it uses "ts" as its host name.

By default, the protocol translator uses its name to form a host configuration file name. To form this file name, the protocol translator converts its name to all lowercase letters, removes all domain information, and appends "-confg."

Example:

Suppose the protocol translator has the name "chaff" and the Internet address 192.31.7.18. The "network-confg" file would include this line:

```
ip host chaff 192.31.7.18
```

After loading the "network-confg" file, the protocol translator would try to load the configuration file "chaff-confg" using TFTP. This configuration file would contain commands specific to "chaff," such as **interface** and **line** configuration commands.

The protocol translator uses TFTP to load configuration files. By default, the protocol translator uses an Internet address of all ones to broadcast TFTP Read Request messages. However, many hosts use an old style of broadcast address consisting of all zeros. You can change protocol translator operation to accommodate hosts using the old style of broadcast address using the **ip broadcast-address** interface subcommand.

If the protocol translator fails to load a configuration file during startup, it tries again every 10 minutes until a host provides the requested files. With each (failed) attempt, the protocol translator displays a message on the console terminal.

Example:

If the protocol translator is unable to load the file named "network-confg," it displays the message:

```
Booting network-confg ... [timed out]
```

To end these file load attempts, enter the **no service config** command on the console terminal.

Configuring the System

This section contains procedures and command descriptions for configuring the global system characteristics, host name, and passwords. This section also describes configuring system security and system management functions.

Note: The global configuration commands **banner** (all versions), **busy-message**, **lockable**, and **state-machines** are only applicable in two-step translations and when using the system console.

Setting the Host Name

The **hostname** global configuration command specifies the host name for the network server, which is used in prompts and default configuration file names. To specify or modify the host name for the network server, use **hostname** global configuration command. The command syntax is:

hostname *name*

The argument *name* is the new host name for the network server and is case-sensitive. The factory assigned default host name is *CPT* or *GATEWAY.* However, you are required to enter a new name during initial setup.

Example:

This command changes the host name to *sandbox*.

```
hostname sandbox
```

Setting Banner Messages

A banner is the message that the EXEC command interpreter displays whenever a user starts any EXEC process or activates a line.With the protocol translator, this command applies only to two-step translations.

The general form of the banner command is:

banner {motd | exec | incoming} *d text d*

The argument *d* specifies a delimiting character of your choice. The argument *text* specify the message to be shown on the screen whenever an interface line is activated.

Follow the **banner** command with one or more blank spaces and then type the delimiting character, one or more lines of text *text*, terminating the message with the second occurrence of the delimiting character.

Example:

The following example uses the # as a delimiting character:

```
banner motd #
Building power will be off from 7:00 AM until 9:00 AM this coming Tues-
day.
#
```

Note: You cannot use the delimiting character in the banner message. In the example, the # symbol cannot be used in the message.

Displaying a Message-of-the-Day Banner

To specify a general-purpose message-of-the-day type banner, use the **banner motd** global configuration command. The command syntax is:

banner motd *d text d*

Note: The command **banner** *d text d is* equivalent to the command **banner motd** *d text d*, except that the banner is now displayed on incoming connections also.

This message-of-the-day type banner is displayed whenever a line is activated or when an incoming Telnet connection is created. When a new **banner motd** command is added to the configuration, it overwrites the existing **banner** command (no keyword specified). Similarly, if a **banner** command is added to the configuration, any exiting **banner motd** command is overwritten.

Displaying a Banner with an EXEC Process

To be able to display a message when an EXEC process is created, use the **banner exec** global configuration command. The command syntax is:

banner exec *d text d*

This specifies a message to be displayed on when an EXEC process is created (line activated, or incoming connection to VTY.)

Displaying an Incoming Message Banner

To display an incoming message to a particular terminal line, use the **banner incoming** global configuration command.

An “incoming” connection is one initiated from the Ethernet side of the protocol translator.

The command syntax is:

banner incoming *d text d*

This specifies a message to be displayed on incoming connections to particular terminal lines.

Note: Messages are never displayed on incoming stream type connections, as they might interfere with printer daemons.

The EXEC banner can be suppressed on certain lines using the **no exec-banner** line subcommand. This line should *not* display the EXEC or MOTD banners when an EXEC is created.

Example:

Suppose we want a message to tell that the box is going to be reloaded with new software tonight. The following example shows how to use the **banner** global configuration command and **no exec-banner** line subcommand to accomplish this setting.

```
! Both messages are inappropriate for the VTYs.
line vty 0 4
no exec-banner
!
banner exec /
This is Cisco Systems training group protocol translator.

Unauthorized access prohibited.
/
!
banner incoming /
You are connected to a Hayes-compatible modem.

Enter the appropriate AT commands.
Remember to reset anything to change before disconnecting.
/
!
banner motd /
The protocol translator will go down at 6pm for a software upgrade
/
```

Setting the System Buffers

In normal system operation, there are several pools of different sized buffers. These pools grow and shrink based upon demand. Some buffers are temporary and are created and destroyed as warranted. Other buffers are permanently allocated and cannot be destroyed. The **buffers** command allows a network administrator to adjust initial buffer pool settings, as well as the limits at which temporary buffers are created and destroyed. It is normally not necessary to adjust these parameters; do so only after consulting with Cisco support personnel. Improper settings could adversely impact system performance. The full syntax of this command follows:

[no] buffers {small | middle | big | large | huge} {*permanent* **|** *max-free* **|** *min-free* **|** *initial***}** *number*

The first argument to the command is the name of the buffer pool; the name denotes the size of buffers in the pool—small, big, huge, etc. The default number of the buffers in a pool is determined by the hardware configuration, and can be displayed with the EXEC **show buffers** command.

The second argument specifies the buffer management parameter to be changed, and can be one of the following arguments:

- *permanent*—The number of permanent buffers that the system tries to allocate. Permanent buffers are normally not deallocated by the system.
- *max-free*—The maximum number of free or unallocated buffers in a buffer pool.
- *min-free*—The minimum number of free or unallocated buffers in a buffer pool.
- *initial*—The number of additional temporary buffers which should be allocated when

the system is reloaded. This can be used to insure that the system has necessary buffers immediately after reloading in a high traffic environment.

- *number*—Specifies the number of buffers to be allocated.

The *no buffers* command with appropriate keywords and arguments restores the default buffer values.

Examples:

In the following example, the system will try to keep at least 50 small buffers free.

```
buffers small min-free 50
```

In this example the system will try to keep no more than 200 medium buffers free.

```
buffers medium max-free 200
```

With the following command, the system will try to create one large temporary extra buffer, just after a reload:

```
buffers large initial 1
```

In this example the system will try to create one permanent huge buffer:

```
buffers huge permanent 1
```

To display statistics about the buffer pool on the system, use the command **show buffers**. For more information, refer to the section on "Monitoring System Processes" in the "Managing the System" chapter.

Displaying a Host Failed Message

To display a specific message when a connection fails with a specified host, use the **busy-message** configuration command. With the protocol translator, this command is only useful in two-step translations. The command syntax is:

busy-message *hostname d message d*
no busy-message *hostname*

The **busy-message** command defines a message that the protocol translator displays whenever an attempt to connect to the specified host fails. This command applies only to Telnet connections.

The argument *hostname* is the name of the host. Follow *hostname* with one or more blank spaces and a delimiting character (*d*) you choose. Then, type one or more lines of text (*message*), terminating the message with the second occurrence of the delimiting character.

The **no busy-message** command disables the busy-message from displaying on the specified host.

Example:

The following example uses the # as a delimiting character. The message will be displayed on the terminal whenever an attempt to connect to the dross host fails:

```
busy-message dross #
Can not connect to host. Contact the computer center.
#
```

Note: You cannot use the delimiting character in the busy message.

Defining a Login String

To send a specific string after a successful connection, use the **login-string** global configuration command. With the protocol translator, this command is only useful in two-step translations. The command syntax is:

login-string *hostname d message* **[%***sec***p] [%***sec***w] [%b]** *d*
no login-string *hostname*

This command defines a string of characters that the protocol translator sends to a host after a successful connection attempt. The argument *hostname* is the name of the host to receive the message. Follow *hostname* with one or more blank spaces and a delimiting character (*d*) you choose. Then type one or more lines of text *message*, terminating the message with the second occurrence of the delimiting character.

The **%***sec***p** option sets a pause in seconds.

The **%***sec***w** option prevents users from issuing commands or keystrokes during a pause.

The **%b** option sends a Break character.

To use a percent sign in the login string, precede it with another percent sign; that is, type the characters "%%."

The **no login-string** subcommand removes the login string. This command applies only to rlogin and Telnet sessions.

Note: You cannot use the delimiting character in the login message.

To insert pauses into the login string, embed a percent sign (%) followed by the number of seconds to pause and the letter "p."

Example:

In the following example the value "%5p" causes a five-second pause:

```
login-string office #ATDT 555-1234
%5p hello
#
```

Locking the Terminal

To enable the EXEC Command **lock**, use the **lockable** global configuration command. With the protocol translator, this command is only useful in two-step translations. The command syntax is:

[no] lockable

This command allows a terminal to be temporarily "locked" by the EXEC command **lock**.

The **no lockable** command reinstates the default, which does not allow the terminal to be locked.

Setting Up State Machines for TCP

State machines, in general, allow control of processes based upon a set of inputs. The current state of the device determines what will happen next given an expected input. The state-machine commands configure the protocol translator to search for and recognize a particular sequence of characters, then cycle through a set of states. The user defines these states using the state-machine command; up to eight states can be defined. (Think of each state as a step the protocol translator takes based upon the assigned configuration commands, and the type of information received.)

The purpose of a state machine is to allow packet dispatch based upon a sequence of characters, instead of a character. The **dispatch-character** commands (described in "State Machine Specification" section later in this chapter) enable packets to be buffered, then transmitted upon receipt of a character. The **state-machine** command allows packets to be buffered, then transmitted upon receipt of a sequence of characters. This allows for packet transmission by pressing a function key, which is typically defined as a sequence of characters ("Esc [A", as an example).

The protocol translator code supports user specified state machines for determining whether data from an asynchronous port should be sent to the network. This is an extension of the concept of dispatch-character, and allows (for example) the equivalent of multicharacter dispatch strings.

To specify the transition criteria for the state of a particular state machine, use the **state-machine** global configuration command. With the protocol translator, this command is only useful in two-step translations. The command syntax is:

state-machine *name state firstchar lastchar nextstate* **[transmit] [delay]**

The *argument name* is the user-specified name for the state machine (used in the **dispatch-machine** line subcommand). There can be any number of state machines specified by the user, but each line can only have a single state machine associated with it.

The argument *state* defines which state is being modified. There are a maximum of eight states per state machine. Lines are initialized to state 0, and return to state 0 after a packet is transmitted.

The arguments *firstchar* and *lastchar* specify a range of characters. If the state machine is in the indicated state, and the next character input is within this range, the process goes to the specified next state. Full 8-bit character comparisons are done, so the maximum value is 255. Take care that the line is configured to strip parity bits (or not generate them) or duplicate the low characters in the upper half of the space.

The argument *nextstate* defines the state to enter if the character is in the specified range.

Specifying the **transmit** keyword causes the packet to be transmitted, and the state machine to be reset to state 0. Characters that occur that have not been explicitly defined to have a particular action return the state machine to state 0.

The optional **delay** keyword specifies that the destination state is transitory. If no additional input is received, the packet will be sent after 100 mS, and the state reset to 0.

This command is entered with the **dispatch-machine** line subcommand, which defines the line on which the state machine is effective.

For more information on setting the state machine for TCP, see the section "State Machine Specification." The following two examples show two different ways to setup state machines:

Following are two examples that show two different ways to set up state machines.

Example 1:

The first example is on the same line as an asynchronously-based packet format. A packet is terminated by a sequence DLE, ETX, CHKSUM, (16, 3, some number) DLE can occur in the packet itself if escaped by another DLE. (In particular, this means that the sequence DLE, DLE, ETX, X should not terminate the packet.)

Use a long dispatch time-out to ensure that large packets can be generated, and to prevent the line from getting permanently stuck in the event of lost data.

```
!If we see the first DLE, go to state 1 (from state 0)
!
state-machine packet 0 16 16 1
!
! If we see ETX after the DLE, go to state 2, otherwise (including
! another DLE) return to state 0
!
state-machine packet 1 3 3 2
```

```
!
! In state 2, receipt of the checksum causes the packet to be sent
!
state-machine packet 2 0 255 transmit
!
! Add this state machine to the appropriate lines
!
line 1 20
dispatch-machine packet
```

Example 2:

The second example attempts to insure that the characters from the function keys on an ANSI terminal are all lumped together into a single packet. Because of this, systems that attempt to distinguish between function keys and the same bytes typed individually do not become confused by variable network delays. An ANSI function key usually generates "Esc [*random* upper-case-alpha":

```
! Recall that the default is to remain in state 0 without
! transmitting anything. We want normal type-in to be transmitted
! immediately
!
state-machine function 0 0 255 transmit
!
! Except for "escape," which starts waiting for the rest of a
! function key.  However, if the user types "escape" we want it
! to be transmitted pretty soon.  This is what the "delay"
! keyword does.
! state-machine function 0 27 27 1 delay
!
! Again, "esc foo" should transmit immediately, unless foo is "["
!
state-machine function 1 0 255 transmit
state-machine function 1 91 91 2 delay
!
! Finally, we want to collect perhaps many characters in state 2,
! Until we run into an upper case alphabetic character, or the
! line stays idle for a while.
!
state-machine function 2 0 255 2 delay
state-machine function 2 65 90 transmit
```

Setting Configuration File Specifications

This section describes the global configuration commands used to configure both the network and the host configuration files.

To configure the system and file specifications, you need to be in the configuration command mode. To enter this mode, use the EXEC **configure** command.

The **boot** command changes default file names and may specify a server host for netbooting configuration files, boot image files, and the size buffer to configure for netbooting a host or network configuration file.

Changing the Network Configuration File

The network configuration file contains commands that apply to all network servers and protocol translators on a network. The default name of this file is "network-config. "Refer to the "Configuration Methods" section earlier in this chapter for more details. To change the name of the "network-config" file use the **boot network** global configuration command. The command syntax is:

boot network *filename* [*address*]
no boot network [*filename address*]

The keyword **network** changes the network configuration file from "network-confg."

The argument *filename* is the new name for the network configuration file.

The argument *address* is the new broadcast address. If you omit the argument *address,* the protocol translator uses the default broadcast address of "255.255.255.255." If you use *address,* you can specify a specific network host or a subnet broadcast address.

Note: If you specify more than one of these in a configuration, the second one that appears will take precedence.

Changing the Host Configuration File

The host configuration file contains commands that apply to one network server in particular. To change the host configuration file name, use the **boot host** global configuration command. The command syntax is:

boot host *filename* [*address*]
no boot host [*filename address*]

The keyword **host** changes the host configuration file name to a name you specify in the *filename* argument.

By default, the protocol translator uses its name to form a host configuration file name. To form this name, the protocol translator converts its name to all lowercase letters, removes all domain information, and appends "-confg." By default, the host file name is "cpt-confg."

Obtaining the System Image Over the Network

A protocol translator can execute its system software stored in ROM, or it can be configured to load a newer version across the network. Network loading is only supported on systems with at least 4 Mbytes of memory (for example, CSC/3 or expanded IGS). The system uses the default file name *cisconn-cpu*, where "nn" is a value from the configuration register and "cpu" is the processor board in the system. You can override this with the **boot system** command.

boot system *filename* **[***address***]**
no boot system [*filename address***]**

The keyword **system** indicates this is a request to set the system image filename. In this case, the argument *filename* is the file name of the operating software to load, and the argument *address* is the address of the network host holding that file.

To use the non-volatile memory option to specify netbooting, place a **boot system** command in the non-volatile memory. This command is used to specify both the file name of the operating software to load, and the Internet address of the server host holding that file.

Example:

This command uses the non-volatile memory to specify the file name `/usr/local/tftpdir/Cisco.ts2` to load, and the Internet address `192.7.31.19` of the server host holding that file:

```
boot system /usr/local/tftpdir/Cisco.ts2 192.7.31.19
```

To remove a file-name-and-address pair from the list, use the **no** form of the **boot** command syntax.

The **boot system** command overrides the processor configuration register setting unless the register specifies the use of default (ROM) operating software. Therefore, to permit netbooting, set the configuration register bits on the processor card to any pattern other than 0-0-0-0 or 0-0-0-1.

Specifying a Boot File Buffer Size

Normally, the protocol translator uses a buffer the size of the system non-volatile memory to hold configuration commands read from the network. You can increase this size if you have a very complex configuration using the **boot buffersize** command. The command syntax is:

boot buffersize *bytes*
no boot buffersize *bytes*

The argument *bytes* specifies the size of the buffer to be used. By default it is the size of the non-volatile memory, and there is no minimum or maximum size that may be specified.

The EXEC commands **write terminal** and **write network** use the information specified by the **buffersize** keyword when performing their functions (refer to the section "Configuration Methods" earlier in this chapter for more information about these EXEC commands.)

You can issue multiple instances of all variations of the **boot** command, including the **no boot** forms. This feature can be useful for removing configuration files.

Configuring Multiple Instances of the Boot Commands

You can configure multiple instances of the **boot** commands. When issued, each command is executed in order and so can be used to begin a systematic search or to build a specific list. For example, you can issue multiple **boot** commands to build an ordered list of configuration-file-name-and-host-address pairs. The protocol translator scans this list until it success-

fully loads the appropriate network or host configuration file or system boot image. In this example, the protocol translator looks first for `fred-confg` on `192.31.7.24` and, if it cannot load that file, then for `wilma-confg` on `192.31.7.19`:

```
boot host /usr/local/tftpdir/fred-confg 192.31.7.24
boot host /usr/local/tftpdir/wilma-confg 192.31.7.19
```

Note: This example uses fictitious file names; the syntax of these file names depends on the TFTP server you are loading the files from.

If the protocol translator cannot find either file, a background process tries at ten-minute intervals (default) to load one or the other of the files.

You may issue multiple instances of all variations of the **boot** command, including the **no boot** forms. This feature can be useful for removing configuration files. To remove a configuration file-name and host-address pair from the list, use the **no** form of the **boot** command syntax.

Establishing Passwords and System Security (TACACS)

This section describes how to configure password protection and terminal access security.

You may set passwords to control access to the privileged command level and to individual lines. Additionally, you can configure a protocol translator to use a special protocol called Terminal Access Controller Access Control System (TACACS) to allow a finer level of control using a server running on a timesharing system. The Defense Data Network (DDN) developed TACACS to control access to its TAC protocol translators; Cisco patterned its TACACS support after the DDN application.

Additional protection by use of access lists may also be required. The use of access lists applies to TCP/IP-based connections. Refer to the "Configuring TCP/IP" chapter for more information.

Establishing the Privileged-Level Password

To assign a password for the privileged command level, use the **enable password** global configuration command. The command syntax is:

enable password *password*

The argument *password* is case-sensitive and specifies the password prompted for in response to the EXEC command **enable.** The *password* argument may contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is also case-sensitive. The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

When you use the **enable** command at the console terminal, the EXEC will not prompt for a password if the privileged mode password is not set. This behavior allows access to the **configure** command to enter configuration command collection mode, so you can set parameters—such as the password for the privileged command level—for other lines. To restrict access to the console line, set a line password as described in the section "Establishing Line Passwords" later in this chapter.

Example:

This example sets the password *secretword* for the privileged command level on all lines, including the console:

```
enable password secretword
```

Specifying a Password

When an EXEC is started on a line with password protection, the EXEC prompts for the password. If the user enters the correct password, the EXEC prints its normal privileged prompt. The user may try three times to enter a password before the EXEC exits and returns the terminal to the idle state.

To specify a password, use the **password** line subcommand. The command syntax looks like this:

password *text*
no password

The *text* argument may contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is also case-sensitive. The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

To enable checking for the password specified by the **password** command, use the line subcommand **login**:

login

Alternatively, to cause the TACACS-style user ID and password checking mechanism to be used instead, use the following subcommand**:**

login tacacs

To disable all password checking, use the command:

no login

The protocol translator prints the message-of-the-day banner before prompting for a password, so the user immediately sees messages such as "no trespassing" notifications. By default, virtual terminals require a password. If you do not set a password for a virtual terminal, it will respond to attempted connections by displaying an error message and closing the connection. Use the **no login** subcommand to disable this behavior and allow connections without a password.

Example:

This example sets the password *letmein* on line 5:

```
line 5
password letmein
login
```

If your network server has the non-volatile memory option, you can "lock yourself out" if you enable password checking on the console terminal line and then forget the line password. To recover from this situation, force the network server into factory diagnostic mode by turning off the network server, insert a jumper in bit 15 of the processor configuration register, (or bit 7 of the processor configuration register in the IGS chassis), and turn on the network server. The processor configuration registers are described in Appendix A of the *Modular Products Hardware Installation and Reference* or the *IGS Hardware Installation and Reference* publications.

When the network server restarts in factory diagnostic mode, it does not read the non-volatile memory, thus avoiding the command to set a password for the console terminal. At the prompt, type **show config** to display the password in the configuration file. Do not change anything in the factory diagnostic mode. To resume normal operation, turn off the network server, remove the jumper from bit 15 (or bit 7) of the configuration register, and turn on the network server again. Log in to the network server with the password which was shown in the configuration file earlier.

Establishing Terminal Access Control

Cisco Systems provides unsupported versions of a standard and an extended TACACS server that run on most UNIX systems available from Cisco through the IP FTP protocol. The servers may be used to create UNIX accounting records to track use of server usage.

What follows are the configuration commands that tailor the behavior of the standard TACACS software running on the Cisco server.

Setting the Server Host Name

The **tacacs-server host** global configuration command specifies a TACACS host. The command syntax is:

tacacs-server host *name*
no tacacs-server host *name*

The argument *name* is the name or Internet address of the host. You can use multiple **tacacs-server host** subcommands to specify multiple hosts. The server will search for the hosts in the order you specify them.

The **no tacacs-server host** global configuration command deletes the specified name or address.

Limiting Login Attempts

The **tacacs-server attempts** global configuration command controls the number of login attempts that may be made on a line set up for TACACS verification. The command syntax is:

tacacs-server attempts *count*
no tacacs-server attempts

The argument *count* is the number of attempts. The default is three attempts.

The **no tacacs-server attempts** global configuration command restores the default.

Example:

This command changes the login attempt to just one try:

```
tacacs-server attempts 1
```

Setting Login Retries

The **tacacs-server retransmit** global configuration command specifies the number of times the server will search the list of TACACS server hosts before giving up. The server will try all servers, allowing each one to time-out before increasing the retransmit count. The command syntax is:

tacacs-server retransmit *retries*
no tacacs-server retransmit

The argument *retries* is the retransmit count. The default is two retries.

The **no tacacs-server retransmit** global configuration command restores the default.

Example:

This command specifies a retransmit counter value of 5 times:

```
tacacs-server retransmit 5
```

Setting the Timeout Intervals

The **tacacs-server timeout** global configuration command sets the interval the server waits for a server host to reply. The command syntax is:

tacacs-server timeout *seconds*
no tacacs-server timeout

The argument *seconds* specifies the number of seconds. The default interval is 5 seconds.

The **no tacacs-server timeout** global configuration command restores the default.

Example:

This command changes the interval timer to 10 seconds:

```
tacacs-server timeout 10
```

Setting the Last Resort Login Feature

If, when running the TACACS server, the TACACS server does not respond, the default action is to deny the request. The **tacacs-server last-resort** global configuration command can be used to change that default. The command syntax is:

tacacs-server last-resort {password|succeed}
no tacacs-server last-resort {password|succeed}

The command causes the network server to request the privileged password as verification, or forces successful login without further input from the user, depending upon the keyword specified, as follows:

- **password**—Allows the user to access the EXEC command mode by entering the password set by the **enable** command
- **succeed**—Allows the user to access the EXEC command mode without further question

---

Note: The **tacacs-server last-resort** subcommand can be useful when it is important to ensure that login can occur. An example of such a condition is when a systems administrator needs to log in in order to troubleshoot TACACS servers which may be down.

The **no tacacs-server last-resort** global configuration command restores the system to the default behavior.

Establishing Privileged-Level TACACS

The following variations of the **enable** command may be used to configure privileged-level command access using the TACACS protocol.

Enabling the Privileged Mode

The **enable use-tacacs** global configuration command is used to enable use of the TACACS protocol for determining whether a user can access the privileged command level. The command syntax is:

[no] enable use-tacacs

If this command is used, the EXEC **enable** command will ask the user for both a new user name and password pair. This pair is then passed to the TACACS server for authentication. If you are using the Extended TACACS, it will also pass any already-existing UNIX user

identification code to the server.

Enabling the Last Resort Login Feature

The **enable last-resort** global configuration command allows the user to specify what happens if the TACACS servers used by the **enable** command do not respond. The command syntax is:

enable last-resort {succeed|password}
no enable last-resort {succeed|password}

The default action is to fail. Use of the keyword changes the action, as follows:

- **succeed**—Allows the user to enable without further question
- **password**—Allows the user to enter the **enable** mode by typing the **enable password**

The **no enable last-resort** global configuration command restores the default.

Configuring Extended TACACS Features

What follows are the configuration commands that tailor the behavior of the extended TACACS server.

Enabling Extended TACACS Mode

The **tacacs-server extended** global configuration command enables an extended TACACS mode. The command syntax is:

tacacs-server extended

This mode provides information about the terminal requests for use in setting up UNIX auditing trails and accounting files for tracking use of protocol translators, terminal servers, and routers. The information includes responses from protocol translators and routers, and validation of user requests. An unsupported, extended TACACS server is available from Cisco Systems using FTP for UNIX users who want to create the auditing programs. Extended TACACS differs from "standard" TACACS in that standard TACACS provides only user name and password information.

TACACS Notification

The **tacacs-server notify** global configuration command causes a message to be transmitted to the TACACS server, with retransmission being performed by a background process for up to five minutes. The terminal user, however, receives an immediate response allowing access to the feature specified. The command syntax is:

tacacs-server notify {connect|enable|logout}

The optional keywords are used to specify notification of the TACACS server whenever

someone does one of the following things:

- **connect**—User makes TCP connections
- **enable**—User enters the **enable** command
- **logout**—User logs out

Login Authentication

The **tacacs-server authenticate** command requires a response from the network or protocol translator to indicate whether the user may perform the indicated action. The command syntax is:

tacacs-server authenticate {connect|enable}

Actions which require a response include the following, specified as optional keywords:

- **connect**—User makes TCP connections
- **enable**—Use of **enable** command

Establishing User Name Authentication

Networks that cannot support a TACACS service may still wish to use a user name-based authentication system. In addition, it may be useful to define "special" user names that get special treatment (for example, an "info" user name that does not require a password, but connects the user to a general purpose information service.)

The software supports these needs by implementing a local **username** global configuration command. The command syntax is:

username *name* **[nopassword | password** *encryptiontype* **password]**
username *name* **[accesslist** *number***]**
username *name* **[autocommand** *command***]**
username *name* **[noescape] [nohangup]**

Multiple **username** commands can be used to specify options for a single user.

The **nopassword** keyword means that no password is required for this user to log in. This is usually most useful in combination with the autocommand keyword.

The **password** keyword specifies a possibly encrypted password for this user name.

The *encryptiontype* argument is a single digit number. Currently defined encryption types are 0, which means no encryption, and 7, which specifies a Cisco-specified encryption algorithm. Passwords entered unencrypted are written out with the Cisco encryption. A password can contain imbedded spaces and must be the last option specified in the **username** command.

The **accesslist** keyword specifies an outgoing access list that overrides the access list specified in the **access class** line configuration subcommand. It is used for the duration of the user's session. The access list number is specified by the *number* argument.

The **autocommand** keyword causes the command specified by the *command* argument to be issued automatically after the user logs in. When the command is complete, the session is terminated. As the command can be any length and contain imbedded spaces, commands using the autocommand keyword must be the last option on the line.

The **nohangup** keyword prevents the protocol translator from disconnecting the user after an automatic command (set up with the **autocommand** keyword) has completed. Instead, the user gets another login prompt.

Examples:

To implement a service similar to the UNIX **who** command, which can be given at the login prompt and lists the current users of the protocol translator, the command takes the following form:

```
username who nopassword nohangup autocommand show users
```

To implement an information service that does not require a password to be used, the command may take the following form:

```
username info nopassword noescape autocommand telnet nic.ddn.mil
```

To implement an ID that will work even if the TACACS servers all break, the command takes the following form:

```
username superuser password superpassword
```

Configuring the Simple Network Management Protocol

Simple Network Management Protocol (SNMP) provides a way to access and set configuration and runtime parameters for the network server. Cisco System's implementation of SNMP is compatible with RFCs 1155, 1157, and 1213. The Cisco Management Information Base (MIB) supports all of RFCs 1155 and 1213, and provides Cisco-specific variables.

A separate document, available in RFC 1212-type format (concise MIB), describes all the Cisco-specific SNMP variables in the Cisco portion of the MIB. It also describe what is required to establish minimum configuration. Contact Cisco Systems to obtain a copy of this document, which includes instructions for accessing the variables using SNMP.

Defining the SNMP Server Access List

To set up an access list that determines which hosts can send requests to the network server, use the **snmp-server access-list** global configuration command. The command syntax is:

snmp-server access-list *list*
no snmp-server access-list *list*

This command sends all traps to the host. The network server ignores packets from hosts that the access list denies.

The argument *list* is an integer from 1 through 99 that specifies an IP access list number. It applies only to the global read-only SNMP agent configured with the command **snmp-server community**.

The **no snmp-server access-list** global configuration command removes the specified access list.

Examples

This command sends traps to all hosts defined by access list 21:

```
snmp-server access-list 21
```

In this example, *any* host using the community string *braves* has SNMP read-only access to the router.

```
snmp-server access-list 1
snmp-server community bluejay rw
snmp-server community braves ro
!
access-list 1 permit 142.111.131.1
access-list 1 deny 0.0.0.0 255.255.255.255
```

This example configuration restricts SNMP read-only access to only the host at 142.111.131.1:

```
snmp-server community bluejay rw
snmp-server community braves ro 1
!
access-list 1 permit 142.111.131.1
access-list 1 deny 0.0.0.0 255.255.255.255
```

In comparing the two examples, consider that the second example restricts read-only access to *only* the host at 142.111.131.1. The first example allows *any* host SNMP read-only access if it has the community string, and disallows SNMP read-only access to all hosts with the community string *braves* except the host 142.111.131.1.

Setting the System Contact String

To set the system contact string (syscontact), use the **snmp-server contact** command. The command syntax is:

snmp-server contact *text*

The *text* argument is a string that specifies the system contact information.

Setting the System Location String

To set the system location string, use the **snmp-server location** command. The command syntax is:

snmp-server location *text*

The *text* argument is a string that specifies the system location information.

Setting the Community String

To set up the community access string, use the **snmp-server community** global configuration command. The command syntax is:

snmp-server community [*string* [**RO** | **RW**] [*list*]]
no snmp-server [**community** [*string*]]

This command enables SNMP server operation on the network server. The argument *string* specifies a community string that acts like a password and permits access to the SNMP protocol.

By default, an SNMP community string permits read-only access (keyword **RO**); use the keyword **RW** to allow read-write access. The optional argument *list* is an integer from 1 through 99 that specifies an access list of Internet addresses that may use the community string.

The **no snmp-server community** global configuration command removes the specified community string or access list.

Example:

This command assigns the string *comaccess* to the SNMP and allows read-only and specifies that Internet address list 4 may use the community string:

```
snmp-server community comaccess RO 4
```

Establishing the Message Queue Length

To establish the message queue length for each TRAP host, use the **snmp-server queue-length** global configuration command. The command syntax is:

snmp-server queue-length *length*

This command defines the length of the message queue for each TRAP host.

The argument *length* is the number of TRAP events that can be held before the queue must be emptied; the default is 10. Once a TRAP message is successfully transmitted, software will continue to empty the queue, but never faster than at a rate of four TRAP messages per second.

Example:

This command establishes a message queue that traps 4 events before it must be emptied:

```
snmp-server queue-length 4
```

Establishing Maximum Packet Size

To establish the maximum packet size, use the **snmp-server packetsize** global configuration command. The command syntax is:

snmp-server packetsize *bytes*

This command allows control over the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply.

The argument *bytes* is a byte count from 484 through 8192. The default is 484.

Example:

This command establish a packet filtering of a maximum size of 1024 bytes:

```
snmp-server packetsize 1024
```

Establishing the Trap Message Recipient

To specify the recipients of trap messages, use the **snmp-server host** global configuration command. The full syntax follows.

snmp-server host *address community-string* [**snmp**] [**tty**]
no snmp-server host *address*

This command specifies which host or hosts should receive trap messages. You need to issue the **snmp-server host** command once for each host acting as a trap recipient.

The argument *address* is the name or Internet address of the host. The argument c*ommunity-string* is the password-like community string set with the **snmp-server community** command.

The optional keywords define which traps are sent, as follows:

- **snmp**—Enables the SNMP traps described in RFC 1157.
- **tty**—Enables the Cisco enterprise-specific trap when a TCP connection closes.

If you do not specify any optional keywords, the sending of all trap types is enabled.

If you specify multiple **snmp-server host** commands for a given host or address, the community string used is the one on the last command line you entered, and the traps sent are a combination of all the optional keywords you specified.

The **no snmp-server host** command removes the specified host.

Examples

This command sends all SNMP traps to 131.108.2.160:

```
snmp-server host 131.108.2.160
```

To turn these trap messages off, use the **no snmp-server host** command:

```
no snmp-server host 131.108.2.160
```

The following example causes all the SNMP traps to be sent to the host specified by the name *cisco.com*. The community string is defined to be *comaccess*.

```
snmp-server host cisco.com comaccess snmp
```

Examples: Specifying Multiple snmp-server host Commands

Suppose the initial configuration is as follows:

```
snmp-server host 131.108.2.3 public snmp
```

You then enter the following configuration command:

```
snmp-server host 131.108.2.3 private
```

This results in the following configuration, which uses the community string you specified last and the trap type **snmp**:

```
snmp-server host 131.108.2.3 private snmp
```

Starting again with the initial configuration, suppose you enter the following command:

```
snmp-server host 131.108.2.3 notpublic tty
```

This results in the following configuration, which uses the community string you specified last and the trap types **snmp** and **tty**:

```
snmp-server host 131.108.2.3 notpublic snmp tty
```

To modify the initial configuration so that only **tty** traps are sent, enter the following commands:

```
no snmp-server host 131.108.2.3
snmp-server host 131.108.2.3 public tty
```

Establishing the TRAP Message Timeout

To define how often to try resending TRAP messages on the retransmission queue, use this global configuration command. The command syntax is:

snmp-server trap-timeout *seconds*

The argument *seconds* sets the interval for resending the messages. The default is set to 30 seconds.

Example:

This command sets an interval of 20 seconds to try resending TRAP messages on the retransmission queue:

```
snmp-server trap-timeout 20
```

Enabling SNMP System Shutdown Feature

Using SNMP packets, a network management tool can send messages to users on virtual terminals and the protocol translator's console. This facility operates in a similar fashion to the EXEC **send** command; however, the SNMP request that causes the message to be issued to the users also specifies the action to be taken after the message is delivered. One possible action is a "shutdown" request.

Requesting "shutdown-after-message" is similar to issuing a **send** command followed by a **reload** command. Because the ability to cause a reload from the network is a powerful feature, it is protected by this configuration command. To use this SNMP message reload feature the device configuration must include the **snmp-server system-shutdown** global configuration command. The syntax of this command is as follows:

[no] snmp-server system-shutdown

The **no snmp-server system-shutdown** option prevents an SNMP system-shutdown request (from an SNMP manager) from resetting the Cisco agent.

To understand how to use this feature with SNMP requests read the document "mib.txt" available by anonymous ftp from ftp.cisco.com. This document is available in RFC 1212-type format. It describes all the Cisco-specific SNMP variables in the Cisco portion of the MIB. It also describes what is required to establish minimum configuration. Contact Cisco Systems to obtain a copy of this document, which includes instructions for accessing the variables using SNMP.

Disabling the SNMP Server

To disable SNMP server operations on the protocol translator after it has been started, use the **no snmp-server** global configuration command. The command syntax is:

no snmp-server

Tailoring Use of Network Services

The **service** command tailors use by the network server of network-based services. Some **service** commands also configure system defaults; refer to the **decimal-tty** command for an example.

The command syntax is as follows:

service *keyword*
no service *keyword*

The argument *keyword* is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on system with non-volatile memory.

- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; enabled by default.
- **domain**—Specifies ip Domain Name System-based host-name-to-address translation; enabled by default.
- **finger**—Allows Finger protocol requests (defined in RFC 742) to be made of the network server; enabled by default. This service is equivalent to issuing a remote **show users** command.
- **ipname**—Specifies the ip IEN-116 Name Server host-name-to-address translation; disabled by default.
- **nagle**—Enables the **Nagle** algorithm for limiting TCP transactions. By default, this function is disabled.

Use the **service nagle** subcommand to enable use of the Nagle algorithm, which limits TCP transmissions by ensuring only one packet is outstanding per TCP connection. This subcommand is useful in some networks with slow links. Refer to RFC 896 for more information about use of the Nagle algorithm.

The **no service** subcommand with the appropriate keyword disables the specified service or function.

Example:

To disable TFTP autoloading of the configuration files, enter this subcommand:

```
service config
```

To disable use of the Nagle algorithm, enter this subcommand:

```
no service nagle
```

Activating the TCP Keepalive Protocol

The TCP keepalive capability allows a protocol translator to detect when the host with which it is communicating experiences a system failure, even if data stops being transmitted (in either direction). This is most useful on incoming connections. For example, if a host failure occurs while talking to a printer, the protocol translator may never notice, since the printer does not generate any traffic in the opposite direction. If keepalives are enabled, they are sent once every minute on otherwise idle connections. If five minutes pass and no keepalives are detected, the connection is closed. The connection will also be closed if the host replies to a keepalive packet with a reset packet. This will happen if the host crashes and comes back up again.

There are two global commands. They have the following syntax:

[no] service tcp-keepalives-in
[no] service tcp-keepalives-out

The global configuration command **service tcp-keepalives-in** enables keepalives on incoming connections (connections initiated by a remote host). The global configuration

command **service tcp-keepalives-out** enables keepalives on outgoing connections (connections initiated by a user of the protocol translator).

The **no service tcp-keepalives-in** and **no service tcp-keepalives-out** commands disable support of the keepalives protocol. This is the default.

The **show tcp** command display includes keepalive statistics. The "wakeups" row shows how many keepalives have been transmitted without receiving any response (this is reset to 0 when a response is received).

Redirecting System Error Messages

By default, the protocol translator sends the output from the EXEC command **debug** and system error messages to the console terminal.

To redirect these messages, as well as output from asynchronous events such as interface transition, to other destinations, use the **logging** configuration command with the appropriate destination options.

These destinations include the console terminal, virtual terminals, internal buffers, and UNIX hosts running a syslog server; the syslog format is compatible with 4.3 BSD UNIX.

The following sections describe how to implement these redirection options on the protocol translator.

Enabling Message Logging

To configure the logging of messages, you need to be in the configuration command collection mode. To enter this mode use the EXEC command **configure** at the EXEC prompt. (Refer to the section on "Using the Configure Command" earlier in this chapter for more details on this process.)

Next you enable the message logging using the **logging on** command. The command syntax is:

logging on
no logging on

This command enables message logging to all destinations except the console. This behavior is the default.

The **no logging on** command enables logging to the console terminal only.

Logging Messages to an Internal Buffer

The default logging device is the console; all messages are displayed on the console unless otherwise specified.

To log messages to an internal buffer, use the **logging buffered** command. The command syntax is:

logging buffered
no logging buffered

This command copies logging messages to an internal buffer instead of writing them to the console terminal.

The buffer is circular in nature, so newer messages overwrite older messages.

To display the messages that are logged in the buffer, use the EXEC command **show logging**. The first message displayed is the oldest message in the buffer.

The **no logging buffered** command cancels the use of the buffer and writes messages to the console terminal, which is the default.

Logging Messages to the Console

To limit messages logged to the console based on severity, use the **logging console** command. The command syntax is:

logging console *level*
no logging console

This command limits the logging messages displayed on the console terminal to messages with a level at or below *level*.

The argument *level* is one of the following keywords, listed here in order from the most severe to the least severe level:

- **emergencies**—System unusable
- **alerts**—Immediate action needed
- **critical**—Critical conditions
- **errors**—Error conditions
- **warnings**—Warning conditions
- **notifications**—Normal but significant conditions
- **informational**—Informational messages only
- **debugging**—Debugging messages

The default is to log messages to the console at the **warnings** level.

The **no logging console** command disables logging to the console terminal.

Example:

This example limits logging of messages to the console of level **warnings** or below:

```
logging console warnings
```

Logging Messages to Another Monitor

To limit messages logged to the terminal lines (monitors) based on severity, use the **logging monitor** command. The command syntax is:

logging monitor *level*
no logging monitor

This command limits the logging messages displayed on terminal lines other than the console line to messages with a level at or above *level*.

The argument *level* is one of the keywords described for the **logging console** command in the section on "Logging Messages to the Console."

To display logging messages on a terminal, use the privileged EXEC command **terminal monitor.**

The **no logging monitor** command disables logging to terminal lines other than the console line.

Logging Messages to a UNIX Syslog Server

To log messages to syslog server host, use the **logging** command with the appropriate Internet address. The command syntax is:

logging *host*
no logging *host*

The **logging** command identifies a syslog server host to receive logging messages.

The argument *host* is the name or the Internet address of the host to be used as a syslog server.

By issuing this command more than once, you build a list of syslog servers that receive logging messages.

The **no logging** command deletes the syslog server with the specified address from the list of syslogs.

To limit messages logged to the syslog servers based on severity, use the **logging trap** command. The command syntax is:

logging trap *level*
no logging trap

The **logging trap** command limits the logging messages sent to syslog servers to only the messages with a level at or above *level*.

The argument *level* is one of the keywords described for the **logging console** command in the "Logging Messages to the Console" section earlier in this chapter.

The **no logging trap** command disables logging to syslog servers.

Current software generates four categories of syslog messages:

- Error messages about software or hardware malfunctions, displayed at the **errors** level

- Output from the **debug** commands, displayed at the **warnings** level
- Interface up/down transitions and system restart messages, displayed at the **notifications** level
- Reload requests and low-process stack messages, displayed at the **informational** level

The EXEC command **show logging** displays the addresses and levels associated with the current logging setup. The command output also includes ancillary statistics.

Example:

To set up the syslog daemon on a 4.3 BSD UNIX system, include a line such as the following in the file /etc/syslog.conf:

```
local7.debugging                    /usr/adm/logs/tiplog
```

The `local7` keyword specifies the logging facility to be used.

The `debugging` argument specifies the syslog level. (Refer to the previous *level* arguments list for other arguments that can be listed.)

The UNIX system sends messages at or below this level to the file specified in the next field. The file must already exist, and the syslog daemon must have permission to write to it.

Configuring the Console and Virtual Terminal Lines

This section describes the commands for configuring terminals and their characteristics.

To configure your console and virtual terminal lines you need to be in the configuration command collection mode. To enter this mode use the EXEC command **configure** at the EXEC prompt. (Refer to the section on "How to Use the Configure Command" earlier in this chapter for detailed instructions.)

Starting Line Configuration

To start configuring a terminal line, use the **line** command. This command identifies a specific line for configuration and starts the line configuration command collection. The command syntax is:

line [*type-keyword***]** *first-line* **[***last-line***]**

This command can take up to three arguments: a keyword, a line number, or a range of lines numbers.

The optional argument *type-keyword* specifies the type of line to be configured, it is one of the following keywords:

- **console**—The console terminal line.
- **aux**—Auxiliary line, described in the following section.

- **vty**—A virtual terminal for remote console access. The protocol translator host can support 100 virtual terminals for access by incoming Telnet, LAT, MOP, or PAD connections. The IGS and CRM support 32 virtual terminals if concurrently routing and/or bridging.

When the line type is specified, the argument *first-line* is the relative number of the terminal line (or the first line in a contiguous group) you want to configure. Numbering begins with 0 (zero).

The optional argument *last-line* is the relative number of the last line in a contiguous group you want to configure.

If you omit *type-keyword*, then *first-line* and *last-line* are absolute rather than relative line numbers. To display absolute line numbers, use the EXEC command **systat.**

The protocol translator displays an error message if you do not specify a line number.

Note: Line numbers are in decimal form on protocol translators.

The **line** command enables you to easily configure a large group of lines all at once. After you set the defaults for the group, you can use additional **line** commands and subcommands to set special characteristics, such as location, for individual terminal lines.

Once in line configuration command collection mode, you can enter the line subcommands described in the rest of this section. The line configuration command collection mode ends when you enter a command that is not a line subcommand, or when you type Ctrl-Z.

Example:

The following sample command starts configuration for the first five virtual terminal lines, 0 through 4:

```
line vty 0 4
```

Configuring the CPU Auxiliary Port

The **line** command with the keyword **aux** enables use of an auxiliary RS-232 DTE port available on all processor cards. The command syntax is:

line aux 0

This port can be used to attach to an RS-232 port of a CSU/DSU or protocol analyzer. Remote monitoring of that port can be performed by connecting to the TCP port whose number is 2000 decimal plus the line number of the auxiliary port. For example, if the auxiliary port was line 1 (obtained from the EXEC command **show users all**), then the TCP port would be 2001. A special cable must be ordered from Cisco Systems to use the auxiliary port. Refer to your respective *Hardware Installation and Reference* publication for more information about this cable.

When specifying the auxiliary port, address it as line aux 0, as in this sample:

```
line aux 0
```

The auxiliary ports assert DTR only when a Telnet connection is established. The console port does not use RTS/CTS handshaking for flow control.

To configure the auxiliary port to support an EXEC process use the **exec** line subcommand. To allow the port to be configured as an auxiliary port, configure it in the following manner:

```
line aux 0
exec
```

No modem control signals are supported on this line. If an auto-answer modem is configured on the line, you must dial up, log in, then hang up. The EXEC is still present and may be used by the next person that dials into the number. The DTR signal will be active whenever an EXEC is configured on the auxiliary port.

Establishing Line Passwords

When an EXEC is started on a line with password protection, the EXEC prompts for the password. If the user enters the correct password, the EXEC prints its normal prompt. The user may try three times to enter a password before the EXEC exits and returns the terminal to the idle state.

To specify a password, use the **password** line subcommand. The command syntax is as follows:

password *text*
no password

The *text* argument may contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is also case-sensitive.

The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

The **no password** line subcommand removes the password.

Example:

This example sets the virtual terminal line 4 password to “secret”:

```
line vty 4
login
password secret
```

Specifying a Vacant Terminal Banner

The protocol translator displays a message on the console or terminal when there is no active EXEC. This message, referred to as the *vacant message*, is different from the banner message displayed when an EXEC process is activated. To turn the vacant message banner on or off,

use the **vacant-message** line configuration subcommands. The commands syntax are:

vacant-message
vacant-message *d message d*
no vacant-message

The **vacant-message** subcommand enables the banner to be displayed on the screen of an idle terminal.

The **vacant-message** subcommand without any arguments causes the default message to be displayed.

If a banner is desired, follow the **vacant-message** subcommand with one or more blank spaces and a delimiting character *(d*) of your choice. Then type one or more lines of text *message,* terminating the text with the second occurrence of the delimiting character.

Example:

This example turns on the system banner and display a message:

```
line 0
vacant-message #
                Welcome to Cisco Systems, Inc.
        This is the console terminal of the protocol translator Gross.
#
```

The **no vacant-message** line configuration subcommand suppresses a banner message.

Setting a "Lines in Use" Message

The protocol translator allows you to define a custom error message to be displayed when an incoming connection is attempted and lines are in use. If you do not define a custom message, when all lines are in use the user will receive the system-generated default message. You can use this additional text to provide the user with further instructions.

The **refuse-message** line subcommand allows you to define this error message. The command syntax is:

[no] refuse-message *d message d*

The argument *d* is a delimiting character of your choice.

The argument *message* is the message you want to show on the terminal.

To type the message, start with a delimiting character, followed by one or more lines of text, and then terminate the text with the second occurrence of the delimiting character. You should not use the delimiting character within the text of the message.

If you define a custom error message using this command, the protocol translator:

- Accepts the connection
- Prints the custom error message
- Clears the connection

Use the **no refuse-message** option to disable this command.

Note: For a rotary group, you only need to define the message for the first line in the group.

Setting the Terminal Type

To set the terminal type, use the **terminal-type** line subcommand. The command syntax is:

terminal-type *terminal-name*
no terminal-type

The **terminal-type** subcommand records, in the argument *terminal-name,* the type of terminal connected to the line for use in Telnet terminal-type negotiation; the Telnet terminal-type negotiation uses *terminal-name* to inform the remote host of the terminal type. Text is used by TN3270 for display management.

The **no terminal-type** subcommand removes any information about the type of terminal.

Setting the Terminal Location

To set the location of the terminal, use the **location** line subcommand. The command syntax is:

location *text*
no location

The **location** subcommand enters information about the terminal location and/or status.

The argument *text* is the desired description. The description appears in the output of the EXEC command **systat.**

The **no location** subcommand removes this information.

Setting the Line in Insecure Location

To set the line as in an insecure location, use the **insecure** line subcommand. The command syntax is:

insecure
no insecure

This information is presently only used by the LAT software, which reports such connections as “dialup” to remote systems.

In previous versions of Cisco software, any line which used modem control was reported as “dialup” through the LAT protocol; this allows more direct control.

Setting the Screen Length and Width

To set the terminal screen length, use the **length** line subcommand. The command syntax is:

length *screen-length*

The argument *screen-length* is the number of lines on the screen.

The network server uses this value to determine when to pause during multiple-screen output.

The default length is 24 lines. A value of 0 (zero) disables pausing between screens of output.

Note: Not all commands recognize the configured screen length. For example, the **show terminal** command assumes a screen length of 24 lines or more.

To set the terminal screen width, use the **width** line subcommand. The command syntax is:

width *columns*

This subcommand sets the number of characters *columns* on a single line of the attached terminal.

The default is 80 columns. The rlogin protocol uses *columns* to set up terminal parameters on a remote UNIX host.

Note: Hosts can learn the values for both length and width specified with these commands.

Setting the Escape Character

To define or reinstate the default escape character, use the **escape-character** line subcommand. The command syntax is:

escape-character *decimal-number*
no escape-character

The **escape-character** subcommand defines the escape character.

The argument *decimal-number* is either the ASCII decimal representation of the character or a control sequence (Ctrl-E, for example).

The default escape character is Ctrl-^.

Note: The Break key may not be used as an escape character on the console terminal,

because the operating software interprets Break as an instruction to halt the system.

The **no escape-character** subcommand sets the escape character to Break.

Setting the Disconnect Character

To define the disconnect character, use the **disconnect-character** command. The command syntax is:

disconnect-character *decimal-number*
no disconnect-character

This subcommand defines the character you type to end a session with the protocol translator.

The argument *decimal-number* is the ASCII decimal representation of the session-disconnect character.

The Break character is represented by 0 (zero); NUL cannot be represented. By default, the no session-disconnect character is set.

To use the session-disconnect character in normal communications, precede it with the escape character.

The **no disconnect-character** subcommand removes the disconnect character.

Setting the Flow Control

To set the flow control, use the **flowcontrol** subcommand. The command syntax is:

flowcontrol { none | software [in | out]}

The **flowcontrol** subcommand sets the method of data flow control between the terminal or other serial device and the protocol translator.

The keyword **software** sets software flow control.

An additional keyword specifies the direction: **in** causes the protocol translator to listen to flow control from the attached device, and **out** causes the protocol translator to send flow control information to the attached device. If you do not specify a direction, both are assumed.

For software flow control, the default stop and start characters are Ctrl-S and Ctrl-Q (XOFF and XON).

By default, no flow control method is set for a line.

Note: This feature is supported by the protocol translator in that the protocol translator

accepts appropriate requests sent using LAT, Telnet, or PAD protocols.

Setting Character Padding

To set the padding on characters, use the **padding** command. The command syntax is:

padding *decimal-number count*
no padding *decimal-number*

This subcommand sets padding for a specified output character.

The argument *decimal-number* is the ASCII decimal representation of the character.

The argument *count* is the number of NUL bytes sent after that character.

Example:

This example of the **padding s**ubcommand causes a Return (ASCII character *-13) with 25 NUL bytes:

```
padding 13 25
```

The **no padding** subcommand removes padding for the specified output character.

Defining the Transport Protocol for a Specific Line

To determine the protocols that you will allow on individual lines and to select preferred (or default) protocols for these lines, use the **transport** line configuration subcommands.

There are three **transport** commands: **transport output**, **transport input**, and **transport preferred**.

The **transport output** line subcommand has the following syntax:

transport output [telnet] [lat] [pad] [rlogin] [none]

This line subcommand specifies which protocols may be used for output on a line.

The **transport input** line subcommand has the following syntax:

transport input [telnet] [lat] [pad] [none]

This line subcommand controls which protocols may be used to connect to a port.

The **transport preferred** line subcommand has the following syntax:

transport preferred [telnet|lat|rlogin|pad|none]

This line configuration subcommand specifies the preferred protocol to use when a command does not specify one.

For protocol translators that support LAT, the default protocol is LAT. For those that do not support LAT, the default is Telnet.

The keyword **telnet** specifies the remote terminal protocol called Telnet. It allows a user at one site to establish a TCP connection to a login server at another site.

The keyword **lat** specifies the LAT protocol which is a Local Area Transport protocol used most often to connect protocol translators to DEC hosts.

The keyword **pad** specifies the X.29 protocols used most to connect protocol translators to X.25 hosts.

The **rlogin** keyword specifies the rlogin protocol. The rlogin setting is a special case of Telnet. If an rlogin attempt to a particular host has failed, the failure will be tracked, and subsequent connection attempts will use Telnet instead.

The **none** keyword specifies no protocol. The system normally assumes that any unrecognized command is a host name. If the preferred transport is set to **none**, the system no longer makes that assumption. No connection will be attempted if the command is not recognized.

The following example sets the preferred protocol output on virtual terminal line number 1 to rlogin:

```
!
line vty 1
transport preferred rlogin
!
```

Refer to the "Selecting the Preferred Terminal Transport Protocol" section in the "Managing the System" chapter for more examples illustrating transport protocol specification.

Establishing an Automatic Connection

To establish an automatic connection to a host, use the **autohost** command. The command syntax is:

autohost *connect-arguments*

The **autohost** subcommand causes a line to automatically establish a connection to a host when an EXEC is started.

The argument *connect-argument* is any text appropriate as an argument to the **connect** EXEC command, including the hostname and any switches.

The **autohost** command is typically used to force an automatic call to a particular host when a user connects to a designated line on the terminal server.

Example:

The following example forces an automatic connection to a host named "dustbin" (which could be an IP address). In addition, uucp specifies TCP socket 25, and **/stream** enables a raw TCP stream with no Telnet control sequences.

```
autohost dustbin uucp /stream
```

Configuring Automatic Line Disconnect

To configure the Automatic Line Disconnect feature for the protocol translator, use the following command:

autohangup

This subcommand causes the EXEC to issue the **exit** command when the last connection closes. This subcommand is useful for UNIX UUCP applications that require this behavior, because UUCP scripts cannot issue the **exit** command to hang up the telephone line.

Setting the Dispatch Character and Character Buffer

To set the dispatch character for the packets, use the **dispatch-character** line subcommand. The command syntax is:

dispatch-character *decimal-number1* **[***decimal-number2 . . . decimal-numbern***]**
no dispatch-character *decimal-number1* **[***decimal-number2 . . . decimal-numbern***]**

This line subcommand defines a character that causes the packet to be sent, even if the dispatch timer has not expired.

The argument *decimal-number* is the ASCII decimal representation of the character, such as Return (ASCII character 13) for line-at-a-time transmissions.

The subcommand can take multiple arguments, so you can define any number of characters as the dispatch character.

The **no dispatch-character** subcommand removes the definition of the specified dispatch character.

To set the dispatch timer, use the **dispatch-timeout** line subcommand. The command syntax is:

dispatch-timeout *milliseconds*
no dispatch-timeout *milliseconds*

This subcommand sets the dispatch timer.

The argument *milliseconds* specifies the number of milliseconds the protocol translator waits after putting the first character into a packet buffer before sending the packet. During this interval, more characters may be added to the packet, thus increasing the processing efficiency of the remote host.

Note: The communications response may appear intermittent if the time-out interval is greater than 100 milliseconds and remote echoing is used.

The **no dispatch-timeout** subcommand removes the time-out definition.

The **dispatch-character** and **dispatch-timeout** subcommands together cause the protocol translator to attempt to buffer characters into larger-sized packets for transmission to the remote host. The protocol translator normally dispatches each character as it is typed.

State Machine Specification

To specify the state machine for TCP packet dispatch, use the **dispatch-machine** command. The command syntax is:

dispatch-machine *name*

The *name* argument specifies the name of the state machine which determines when to send packets on the asynchronous line.

Any dispatch characters specified using the **dispatch-character** command are ignored if a state machine is also specified.

When the **dispatch-timeout** command is specified, a packet being built will be sent when the timer expires, and the state will be reset to 0.

If a packet becomes full, it will be sent regardless of the current state. However, the state is not reset. The packet size is dependent on the amount of traffic on the asynchronous line, as well as the dispatch time out. There is always room for 60 data bytes, and if the dispatch-timeout is 100 ms or greater, a packet size of 536 (data bytes) is allocated.

For more examples and information about the state machine, refer to the "Setting Up State Machines for TCP" in an earlier section of this chapter.

Turning the EXEC On or Off

To turn the EXEC on or off, use the **exec** subcommand. The command syntax is:

[no] exec

This subcommand determines whether or not the protocol translator will start an EXEC process on the line. A serial printer, for example, should not have an EXEC started.

By default, the protocol translator starts EXECs on all lines.

Suppressing EXEC Banners

Use the **exec-banner** subcommand to control whether banners are displayed or suppressed. The command syntax is:

[no] exec-banner

This line subcommand determines whether or not the terminal server will display the EXEC banner or message-of-the-day (MOTD) banner when an EXEC is created.

By default, the messages defined with **banner motd** and **banner exec** are displayed on all lines. The **no exec banner** subcommand is useful for temporarily suppressing these banner messages. Specify **exec-banner** to reinstate.

Example:

This example suppresses the banner on virtual terminal lines 0 through 4:

```
line vty 0 4
no exec-banner
```

Setting the EXEC Timeout Intervals

The EXEC command interpreter waits for a specified interval of time until user start input, if no input is detected, the EXEC resumes the current connection, or if no connections exist, it returns the terminal to the idle state and disconnects the incoming session.

To set this interval use the **exec-timeout** subcommand. The command syntax is:

exec-timeout *minutes* [*seconds*]
no exec-timeout

The argument *minutes* is the number of minutes.

The optional argument *seconds* specifies additional time intervals in seconds. The default interval is 10 minutes; an interval of 0 (zero) specifies no time-outs.

The **no exec-timeout** subcommand removes the time-out definition. It is the same as entering `exec-timeout 0`.

Example:

This sample subcommand sets a time interval of 2 minutes, 30 seconds:

```
exec-timeout 2 30
```

This following sample subcommand sets an interval of 10 seconds:

```
exec-timeout 0 10
```

Setting the Interval for Closing a Session

To set the interval for closing the connection when there is no traffic, use the **session-timeout** subcommand. The command syntax is:

session-timeout *minutes*
no session-timeout

This subcommand sets the interval that the protocol translator waits for input or output traffic before closing the connection to a remote computer and returning the terminal to an idle state.

The argument *minutes* specifies the time interval in minutes.

The default interval is 0 (zero), indicating the protocol translator maintains the connection indefinitely.

Example:
The following subcommand sets an interval of 10 minutes:

```
session-timeout 10
```

The **no session-timeout** subcommand removes the time-out definition.

However, you can specify a session time-out on each port.

Setting the Session Limit

To set the maximum number of sessions, use the **session-limit** command. The command syntax is:

session-limit *session-number*
no session-limit

The argument *session-number* specifies the maximum number of sessions.

Setting Input Notification

To enable terminal notification about pending output from other connections, use the **notify** command. The command syntax is:

[no] notify

The **notify** subcommand sets a line to inform a user who has multiple, concurrent Telnet connections when output is pending on a connection other than the current connection.

This subcommand performs the same function as the EXEC command **terminal notify** described in "Changing Terminal Parameters" in the "Managing the System" chapter.

The **no notify** subcommand ends notification.

Configuring Rotary Groups

Connections to the protocol translator made to the next free line in a group of lines, is called a *rotary group.* A line may be in only one rotary group; however, a rotary group may consist of a single line or many lines.

Note: The console line (line 0) may not be in a rotary group.

To define each group of lines, use the **rotary** line subcommand. The command syntax is:

rotary *group*
no rotary

This subcommand adds a line to the specified rotary group.

The argument *group* is an integer you choose between 1 and 100 that identifies the rotary group.

To list the defined rotary groups, use the privileged EXEC command **show line.** For more information, refer to the "Monitoring System Processes" section in chapter 5.

As for connections to an individual line, the remote host must specify a particular TCP port on the protocol translator to connect to a rotary group. The available services are the same, but the TCP port numbers are different. Table 1-2 lists the services and port numbers for both rotary groups and individual lines.

Table 1-2 Services and Port Numbers for Rotary Groups and Lines

Services Provided	Base TCP Port for Rotaries	Base TCP Port for Individual Lines
Telnet protocol	3000	2000
Raw TCP protocol (no Telnet protocol)	5000	4000
Telnet protocol, binary mode	7000	6000

S1603

For example, if Telnet protocols are required, the remote host connects to the TCP port numbered 3000 (decimal) plus the rotary group number. If the rotary group is 13, the corresponding TCP port is 3013.

If a raw TCP stream is required, the port is 5000 (decimal) plus the rotary group number. If rotary group 5 requires a raw TCP (printer) line, it connects to TCP port 5005.

If Telnet binary mode is required, the port is 7000 (decimal) plus the rotary group number.

Rotary Groups on the Protocol Translator Lines

A rotary group can refer to a particular group of virtual terminal lines on a protocol translator. Different line characteristics can be assigned to each rotary group, the most useful being line security and the ability to automatically establish a connection using the **autohost** subcommand.

It is possible, for example, to reserve several virtual terminal lines as special lines and allow the rest to be used for protocol translation. To do this, assign a group of lines, (typically the last virtual terminal lines) to a rotary group with a special password:

Example:

```
interface ethernet 0
ip address 131.108.2.34 255.255.0.0
line vty 0 94
no login
line vty 95 99
login
password managementpassword
rotary 1
ip alias 131.108.2.35 3001
```

Now, when a user connects to the protocol translator's main address, he or she will probably get a standard virtual terminal line and will not have to log in (unless the first 95 virtual terminal lines are already in use). A network manager can always make a Telnet connection to the alias address or rotary port (3001), and access a line that no one else can use.

If the X.25 network supports subaddressing (for IP, the numbers following the first ten in an address), the protocol translator treats any subaddress on incoming PAD calls as a rotary group number. This can be used as in the above example, or combined with the **autohost** command to achieve a primitive form of the "one-step" translation method from PAD to Telnet/rlogin.

Example:

```
! Incoming calls (up to 10) to sub-address 5 get connected to TEST1
line vty 90 99
autohost test1
rotary 5
! Incoming calls to sub-address 6 get connected to ENG2
line vty 80 89
autohost eng2
rotary 6
! Incoming calls to sub-address 7 get connected to Host3
host host3 513 131.104.5.12
line vty 70 79
autohost host3
rotary 7
! All other vtys require a password to get greater freedom
line vty 0 69
login
password goanywhere
```

Console Line Configuration Examples

The following example shows how to enter configuration command collection mode and set a password on the console line:

```
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.
Edit with DELETE, CTRL/W, and CTRL/U; end with CTRL/Z; abort with CTRL/C

line 0
password secret
<Ctrl-Z>
```

The next example shows how to configure a controlled line with no terminal EXEC, with software flow control set in both directions (to and from an attached device). The **dispatch-timeout** subcommand causes the terminal server to wait 100 milliseconds after the first character is put into a packet before sending the packet.

```
line 12
no exec
flowcontrol software
dispatch-timeout 100
<Ctrl-Z>
```

Configuring the Network Interface

This section describes the commands for configuring the system network interface.

The protocol translator displays the name of its interface on the console terminal during startup. To display complete information about the interface, use the EXEC command show interfaces. For more information, refer to the "Monitoring System Processes" section in the "Managing the System" chapter.

Starting the Interface Configuration

In configuration command collection mode, you start configuring an interface with the **interface** command. This command identifies a specific interface for configuration and starts interface configuration command collection.

interface *type unit*

The argument *type* is one of the following generic names for the installed interface: **ethernet** or **serial**.

The argument *unit* is a positive integer that specifies the nth interface of type *type*, starting with 0 (zero) for the first interface of that type. The protocol translator can have only one network interface of each type, so *unit* is always 0.

Example:

To configure unit 0 of the **ethernet** interface, enter this command:

```
interface ethernet 0
```

followed by appropriate interface address subcommand.

Example:

This sample command configures **serial** interface:

```
interface serial 0
```

Follow this command with interface configuration subcommands.

In interface configuration command collection mode, you can enter the interface subcommands described in the rest of this section. The interface configuration command collection mode ends when you enter a command that is not an interface subcommand, or when you type Ctrl-Z.

The protocoDl translator displays the name of its interface on the console terminal during startup. To display complete information about the interface, use the **show interfaces** command. For more information, refer to the "Monitoring System Processes" section in the chapter "Managing the System."

Specifying an Encapsulation Method

The encapsulation method is changed by using the interface configuration subcommand **encapsulation** followed by a keyword that defines the encapsulation method.

encapsulation *encapsulation-type*

For Ethernet and IEEE 802.3 interfaces, the argument *encapsulation-type* is **arpa**. The **arpa** keyword establishes ARPA Ethernet 2.0 encapsulation, which is the default encapsulation method for Ethernet interfaces.

You do not normally need to change the encapsulation method of an Ethernet interface.

The serial interfaces support the following kinds of serial encapsulations:

- High-Level Data Link Control (HDLC)
- HDLC Distant Host (HDH)
- Frame Relay
- Point-to-Point Protocol (PPP)
- Switched Multi-megabit Data Services (SMDS)
- X.25-based encapsulations

For serial interfaces, the *encapsulation-type* argument identifies one of the following serial encapsulation types that Cisco Systems' software supports:

- **bfex25**—Blacker Front End Encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation
- **frame-relay**—Frame Relay
- **hdh**—HDH Protocol
- **hdlc**—Cisco Systems HDLC Protocol
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation

- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-Point Protocol (PPP)
- **smds**—SMDS service
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

Setting Error Count Reset Frequency

The Cisco interface software provides a mechanism for protection against packet overload and resultant recount errors on the MCI interface cards. This mechanism is set using the **error-threshold** interface subcommand. The syntax is as follows:

error-threshold *milliseconds*

The argument *milliseconds* is the frequency at which the error recount will be set. The default is 1000 milliseconds.

Example:

The following example sets the error recount threshold on Ethernet interface 0 to 10000 milliseconds.

```
interface ethernet 0
error-threshold 10000
```

Shutting Down an Interface

To disable an interface, use the **shutdown** subcommand. The command syntax is:

shutdown

This subcommand prevents the transmission of packets on the specified interface.

To restart a disabled interface, use the **no shutdown** subcommand. The command syntax is:

no shutdown

To check whether an interface is disabled, use the **show interface** EXEC command. The command syntax is:

show interface

Global Configuration Command Summary

This section lists all of the global system configuration commands in alphabetical order:

banner *d message d*

Modifies the banner name for the protocol translator.

Follow the **banner** command with one or more blank spaces and any delimiting character (*d*) you choose. Then type one or more lines of text (*message*), terminating the message with the second occurrence of the delimiting character.

To be able to display a message when an EXEC process is created, use the **banner exec** global configuration command.

banner exec *d text d*

This specifies a message to be displayed when an EXEC process is created (line activated, or incoming connection to VTY).

To display incoming message on a particular terminal line, use the **banner incoming** global configuration command.

banner incoming *d text d*

This specifies a message to be displayed on incoming connections to particular terminal lines, for example, lines used for "milking machine" applications.

To specify a general-purpose message-of-the-day type banner, use the **banner motd** global configuration command.

banner motd *d text d*

This is displayed message-of-the-day type banner whenever a line is activated or when an incoming Telnet connection is created.

Note: The command **banner** *d text d* is equivalent to the command **banner motd** *d text d*, except that the banner is displayed on incoming connections also.

[no] boot buffersize *bytes*

Specifies the size of the buffer to be used for netbooting a host or a network configuration file, use the following command:

The argument *bytes* specifies the size of the buffer to be used. By default, it the size of non-volatile memory, and there is no minimum or maximum size that may be specified.

[no] boot host *filename* **[***address***]**

Changes the host configuration file name.

The keyword **host** changes the host configuration file name to a name you specify in the

filename argument.

[no] boot network f*ilename* **[***address***]**

Specifies the name of this file; use the network configuration file.

The keyword **network** changes the network configuration file.

The argument *filename* is the new name for the network configuration file.

The argument *address* is the new broadcast address. If you omit the argument *address,* the protocol translator uses the default broadcast address of "255.255.255.255." If you use *address,* you can specify a specific network host or a subnet broadcast address.

[no] boot system *filename* **[address]**

Specifies the name of an operating system image file to load across the network.

The keyword **system** indicates that the file name and host addresses for booting operating software over the network are in the non-volatile memory. In this case:

The argument *filename* is the file name of the operating software to load, and the argument *address* is the address of the network host holding that file.

[no] buffers {small | middle | big | large | huge} {*permanent* | *max-free* | *min-free* | *initial***}** *number*

Allows a network administrator to adjust initial buffer pool settings and set limits at which temporary buffers are created and destroyed.The first argument denotes the size of buffers in the pool; the default number of the buffers in a pool is determined by the hardware configuration. The second argument specifies the buffer management parameter to be changed, as follows:

- *permanent*—The number of permanent buffers that the system tries to allocate.
- *max-free*—The maximum number of free or unallocated buffers in a buffer pool.
- *min-free*—The minimum number of free or unallocated buffers in a buffer pool.
- *initial*—The number of additional temporary buffers which should be allocated when the system is reloaded.
- *number*—Specifies the number of buffers to be allocated.

The **no buffers** command with appropriate keywords and arguments restores the default buffer values.

[no] busy-message *hostname d message d*

Sets a message that the protocol translator displays whenever an attempt to connect to the specified host fails.

The argument *hostname* is the name of the host. Follow the argument *hostname* with one or more blank spaces and any delimiting character (*d)* you choose. Then type one or more lines of text *message,* terminating the message with the second occurrence of the

delimiting character.

enable password *password*

Assigns a password for the privileged command level.

The argument *password* is case-sensitive and specifies the password prompted for in response to the EXEC command **enable.** [no] enable last-resort {succeed|password}

Allows the user to specify what happens if the TACACS server used by the **enable** command does not respond. The default is to fail.

The keyword **succeed** allows the user to enable without further question.

The keyword **password** allows the user to enable by entering the privileged command level.

The **no enable last-resort** command restores the default.

enable use-tacacs

Use TACACS to check the user ID and password supplied to the EXEC **enable** command.

hostname *name*

Modifies the host name for the protocol translator.

The argument *name* is the new host name for the network server and is case-sensitive. The default host name is "gateway."

[no] lockable

Allows a terminal to be temporarily "locked" by the EXEC command **lock**.

The **no lockable** command reinstates the default, which does not allow the terminal to be locked.

[no] logging *host*

Identifies a syslog server host to receive logging messages.

The argument *host* is the name or the Internet address of the host.

By issuing this command more than once, you build a list of syslog servers that receive logging messages.

The **no logging** command deletes the syslog server with the specified address from the list of syslogs.

[no] logging buffered

Copies logging messages to an internal buffer instead of writing them to the console terminal.

The buffer is circular in nature, so newer messages overwrite older messages.

The **no logging buffered** command cancels the use of the buffer and writes messages to the console terminal, which is the default.

[no] logging console *level*

Limits the logging messages displayed on the console terminal to messages with a level at or below *level*.

The argument *level* is one of the following keywords, listed here in order from the most severe to the least severe level.

- **emergencies**—System unusable
- **alerts**—Immediate action needed
- **critical**—Critical conditions
- **errors**—Error conditions
- **warnings**—Warning conditions
- **notifications**—Normal but significant conditions
- **informational**—Informational messages only
- **debugging**—Debugging messages

The default is to log messages to the console at the **warnings** level.

The **no logging console** command disables logging to the console terminal.

[no] logging monitor *level*

Limits the logging messages displayed on terminal lines other than the console line to messages with a level at or above *level*.

The argument *level* is one of the keywords described for the **logging console** command in the section "Logging Messages to the Console."

The **no logging monitor** command disables logging to terminal lines other than the console line.

[no] logging on

Enables message logging to all destinations except the console. This behavior is the default.

The **no logging on** command enables logging to the console terminal only.

[no] logging trap *level*

Limits the logging messages sent to syslog servers to messages with a level at or above *level*.

The argument *level* is one of the keywords described for the **logging console** command in the section "Logging Messages to the Console" earlier in this chapter.

The **no logging trap** command disables logging to syslog servers.

[no] login-string *hostname d message* **[%***sec***p] [%***sec***w] [%b]** *d*
[no] login-string *hostname*

Defines a string of characters that the protocol translator sends to a host after a successful connection attempt. The argument *hostname* is the name of the host to receive the message. Follow *hostname* with one or more blank spaces and a delimiting character *d* you choose. Then, type one or more lines of text *message*, terminating the message with the second occurrence of the delimiting character. This command applies only to rlogin and Telnet sessions.

The **%***sec***p** option sets a pause in seconds.

The **%***sec***w** option prevents users from issuing commands or keystrokes during a pause.

The **%b** option sends a Break character.

no snmp-server

Disables SNMP server operations on the protocol translator after it has been started.

[no] service *keyword*

Tailors use by the network server of network-based services.

The *keyword* argument specifies the service name.

The argument *keyword* is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on system with non-volatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; enabled by default.
- **domain**—Specifies ip Domain Name System-based host-name-to-address translation; enabled by default.
- **finger**—Allows Finger protocol requests (defined in RFC 742) to be made of the network server; enabled by default. This service is equivalent to issuing a remote **show users** command.
- **ipname**—Specifies the ip IEN-116 Name Server host-name-to-address translation; disabled by defaultt.
- **nagle**—Enables the **Nagle** algorithm for limiting TCP transactions. By default, this function is disabled.

The **no service** subcommand disables the specified service or function.

[no] service tcp-keepalives-in
[no] service tcp-keepalives-out

The command **service tcp-keepalives-in** enables keepalives on incoming connections (connections initiated by a remote host). The command **service tcp-keepalives-out** enables keepalives on outgoing connections (connections initiated by a user of the terminal server).

The **no service tcp-keepalives-in** and **no service tcp-keepalives-out** commands disable support of the keepalives protocol. This is the default.

[no] snmp-server access-list *list*

Sets up an access list that determines which hosts can send requests to the protocol translator. It applies only to the global read-only SNMP agent configured with the command **snmp-server community**.

The argument *list* is an integer from 1 through 99 that specifies an access list.

The **snmp-server access-list** is a global command that sends all traps to the host. The protocol translator ignores packets from hosts that the access list denies.

The **no snmp-server access-list** command removes the specified access list.

[no] snmp-server community [*string* [**RO**|**RW**] [*list*]]

Sets up the community access string. This command enables SNMP server operation on the protocol translator. The argument *string* specifies a community string that acts like a password and permits access to the SNMP protocol.

By default, an SNMP community string permits read-only access (keyword **RO**); use the keyword **RW** to allow read-write access. The optional argument *list* is an integer from 1 through 99 that specifies an access list of Internet addresses that may use the community string.

The **no snmp-server community** command removes the specified community string or access list.

snmp-server contact *text*

Sets the system contact string (syscontact).

The *text* argument is a string that specifies the system contact information.

[no] snmp-server host *address community-string cb* **[snmp|tty]**

Specifies which host or hosts should receive TRAP messages. You need to issue the **snmp-server host** command once for each host acting as a TRAP recipient.

The argument *address* is the name or Internet address of the host. The argument

community-string is the password-like community string set with the **snmp-server community** command.

The optional keyword **snmp** causes all SNMP-type TRAP messages to be sent and sends the initial Cisco-specific RELOAD TRAP message.

The optional keyword **tty** causes TCP connection TRAP messages to be included.

The **no snmp-server host** command removes the specified host.

snmp-server location *text*

Sets the system location string.

The *text* argument is a string that specifies the system location information.

snmp-server packetsize *bytes*

Allows control over the largest SNMP packet size permitted when the SNMP server is receiving a request or generating a reply.

The argument *bytes* is a byte count from 484 through 8192. The default is 484.

snmp-server queue-length *length*

Establishes the message queue length for each TRAP host. It defines the length of the message queue for each TRAP host.

The argument *length* is the number of TRAP events that can be held before the queue must be emptied; the default is 10. Once a TRAP message is successfully transmitted, software will continue to empty the queue, but never faster than at a rate of four TRAP messages per second.

[no] snmp-server system-shutdown

Requesting "shutdown-after-message" is similar to issuing a **send** command followed by a **reload** command. Because the ability to cause a reload from the network is a powerful feature, it is protected by this configuration command. To use this SNMP message reload feature the device configuration must include the **snmp-server system-shutdown** global configuration command.

The **no snmp-server system-shutdown** option prevents an SNMP system-shutdown request (from an SNMP manager) from resetting the Cisco agent.

[no] snmp-server trap-authentication

Enables the network server to send a TRAP message when it receives a packet with an incorrect community string.

The SNMP specification requires that a TRAP message be generated for each packet with an incorrect community string. However, because this action can result in a security breach, the network server by default does not return a TRAP message when it receives

an incorrect community string.

snmp-server trap-timeout *seconds*

Defines how often to try resending TRAP messages on the retransmission queue. The argument *seconds* sets the interval for resending the messages.

The default is set to 30 seconds.

state-machine *name state firstchar lastchar nextstate* **|transmit [delay]**

Specifies the transition criteria for the state of a particular state machine.

The argument *name* is the user-specified name for the state machine (specified by the dispatch-machine line subcommand). There can be any number of state machines specified by the user, but each line can only have a single state machine associated with it.

The argument *state* defines which state is being modified. There are a maximum of eight states per state machine.

The arguments *firstchar* and *lastchar* specify a range of characters. If the state machine is in the indicated state, and the next character input is within this range, go to the specified next state.

The argument *nextstate* defines the state to enter if the character is in the specified range.

Specifying the **transmit** keyword causes the packet to be transmitted, and the state machine to be reset to state 0.

The optional **delay** keyword specifies that the destination state is transitory. If no additional input is received, the packet will be sent after 100 ms, and the state reset to 0.

[no] tacacs-server attempts *count*

Controls the number of login attempts that may be made on a line set up for TACACS verification. The argument *count* is the number of attempts. The default is three attempts.

The **no tacacs-server attempt** subcommand restores the default.

tacacs-server authenticate {connect|slip|enable}

Causes the terminal server to require a response from the TACACS server to indicate whether the user may perform the indicated action. Actions which require a response include the following, specified as optional keywords:

- **connect**—User makes TCP connections
- **enable**—Use of **enable** command

tacacs-server extended

Enables an extended TACACS mode. This mode provides information about the terminal requests for use in setting up UNIX auditing trails and accounting files for tracking use of devices which use extended TACACS. This information includes responses from devices and validation of user requests. An unsupported, extended TACACS server is available from Cisco Systems for UNIX users who want to create the auditing programs.

[no] tacacs-server last-resort {password|succeed}

Causes the network server to request the privileged password as verification, or forces successful login without further input from the user, depending upon the keyword specified.

The keyword **password** allows the user to access the privileged-level command mode by entering the password set by the **enable** command.

The keyword **succeed** allows the user to access the privileged-level command mode without further question.

The **no tacacs-server last-resort** command restores the system to the default behavior.

[no] tacacs-server host *name*

Specifies a TACACS host. The argument *name* is the name or Internet address of the host. You can use multiple **tacacs-server host** subcommands to specify multiple hosts. The server will search for the hosts in the order you specify them.

The **no tacacs-server host** subcommand deletes the specified name or address.

tacacs-server notify {connect|enable|logout}

Causes a message to be transmitted to the TACACS server, when various event occur.

The optional keywords are used to specify notification of the TACACS server whenever someone does one of the following things:

- **connect**—User makes TCP connections
- **enable**—User enters the **enable** command
- **logout**—User logs out

[no] tacacs-server retransmit *retries*

Specifies the number of times the server will search the list of TACACS server hosts before giving up. The server will try all servers, allowing each one to time-out before increasing the retransmit count. The argument *retries* is the retransmit count. The default is two retries.

The **no tacacs-server retransmit** subcommand restores the default.

[no] tacacs-server timeout *seconds*

Sets the time interval that the server waits for a server host to reply. The argument *seconds* specifies the number of seconds. The default interval is 5 seconds.

The **no tacacs-server timeout** subcommand restores the default.

username *name* **[nopassword | password** *encryptiontype* **password]**
username *name* **[accesslist** *number***]**
username *name* **[autocommand** *command***]**
username *name* **[nopassword], [noescape], [nohangup]**

Specifies options for a single user.

The **nopassword** keyword means that no password is required for this user to log in. This is usually most useful in combination with the autocommand keyword.

The **password** keyword specifies a possibly encrypted password for this user name.

The *encryptiontype* argument is a single digit number. Currently defined encryption types are 0, which means no encryption, and 7, which specifies a Cisco-specified weak encryption algorithm. Passwords entered unencrypted are written out with the Cisco encryption. Passwords can contain imbedded spaces and must be the last option specified in the user name command.

The **accesslist** keyword specifies an outgoing access list to be used for the life of the user's login. The access list number is specified by the *number* argument.

The **autocommand** keyword causes the command specified by the *command* argument to be issued automatically after the user logs in. When the command is complete, the session is terminated. As the command can be any length and contain imbedded spaces and such, commands using the autocommand keyword must be the last option on the line.

The **nohangup** keyword prevents the protocol translator from disconnecting the user's connection after an automatic command (set up with the **autocommand** keyword) has completed. Another login prompt is provided to the user.

Line Configuration Subcommand Summary

This section lists all of the line configuration subcommands in alphabetical order:

autohangup

Causes the EXEC to issue the **exit** command when the last connection closes.

autohost *connect-argument*

Causes a line to automatically establish a connection to a host when an EXEC is started.

The argument *connect-argument* is any text appropriate as an argument to the **connect** EXEC command, including the hostname and any switches.

The **autohost** command is typically used to force an automatic call to a particular host when a user connects to a designated line on the terminal server.

[no] disconnect-character *decimal-number*

Defines the character you type to end a session with the protocol translator.

The argument *decimal-number* is the ASCII decimal representation of the session-disconnect character.

[no] dispatch-character *decimal-number1* **[***decimal-number2 . . . decimal-number***]**

Defines a character that causes the packet to be sent, even if the dispatch timer has not expired.

The argument *decimal-number* is the ASCII decimal representation of the character, such as Return (ASCII character 13) for line-at-a-time transmissions.

The **no dispatch-character** subcommand removes the definition of the specified dispatch character.

dispatch-machine *name*

Specifies the name of the state machine to determine when to send packets on the asynchronous line.

The argument *name* specifies the name of the state machine.

[no] dispatch-timeout *milliseconds*

Sets the dispatch timer.

The argument *milliseconds* specifies the number of milliseconds the protocol translator waits after putting the first character into a packet buffer before sending the packet. During this interval, more characters may be added to the packet, thus increasing the processing efficiency of the remote host.

The **no dispatch-timeout** subcommand removes the time-out definition.

[no] escape-character *decimal-number*

Defines the escape character and reinstates the default escape.

The argument *decimal-number* is either the ASCII decimal representation of the character or a control sequence (Ctrl-E, for example).

The default escape character is Ctrl-^.

The **no escape-character** subcommand reinstates the default escape character.

[no] exec-timeout *minutes* **[***seconds***]**

Sets a time interval before returning the terminal to the idle state.

The argument *minutes* is the number of minutes.

The optional argument *seconds* specifies additional interval time in seconds. The default interval is 10 minutes; an interval of 0 (zero) specifies no time-outs.

The **no exec-timeout** subcommand removes the time-out definition. It is the same as entering `exec timeout 0.`

flowcontrol { none | software [in | out]}

Sets the method of data flow control between the terminal or other serial device and the protocol translator.

The keyword **software** sets software flow control.

An additional keyword specifies the direction: i**n** causes the protocol translator to listen to flow control from the attached device, and **out** causes the protocol translator to send flow control information to the attached device. If you do not specify a direction, both are assumed.

For software flow control, the default stop and start characters are Ctrl-S and Ctrl-Q (XOFF and XON).

By default, no flow control method is set for a line.

Note: This feature is supported by the protocol translator in that the protocol translator accepts appropriate requests sent using LAT, Telnet, or PAD protocols.

[no] insecure

Sets the line as in an insecure location.

length *screen-length*

Sets the terminal screen length.

The argument *screen-length* is the number of lines on the screen.

line [*type-keyword***] *first-line* [***last-line***]**

Takes up to three arguments: a keyword, a line number, or a range of line numbers.

The optional argument *type-keyword* specifies the type of line to be configured; it is one of the following keywords:

- **console**—The console terminal line.
- **aux**—Auxiliary line.
- **printer**—A parallel printer line.
- **tty**—A standard asynchronous line.
- **vty**—Virtual terminal for remote console access. The protocol translator host can support 100 virtual terminals for access by incoming Telnet, LAT, MOP or PAD connections.

When the line type is specified, the argument *first-line* is the relative number of the terminal line (or the first line in a contiguous group) you want to configure. Numbering begins with 0 (zero).

The optional argument *last-line* then is the relative number of the last line in a contiguous group you want to configure.

If you omit *type*, then *first-line* and *last-line* are absolute rather than relative line numbers. To display absolute line numbers, use the EXEC command **systat.**

line aux *port-address*

Enables use of an auxiliary RS-232 DTE port available on all processor cards. The *port-address* argument is 1 on Cisco protocol translators.

[no] location *text*

Enters information about the terminal location and/or status.

The argument *text* is the desired description. The description appears in the output of the EXEC command **systat.**

[no] login

Causes the protocol translator to prompt for a password before starting the EXEC process when you have set a password for the line.

The **no login** line subcommand disables the password checking behavior described above and allow connections without a password.

[no] login tacacs

Sets up the TACACS verification mechanism to handle the password checking.

The **no login tacacs** subcommand disables TACACS password checking.

[no] notify

Sets a line to inform a user who has multiple, concurrent Telnet connections when output is pending on a connection other than the current connection.

This subcommand performs the same function as the EXEC command **terminal notify** described in the "Managing the System" chapter.

The **no notify** subcommand ends notification.

[no] padding *decimal-number count*

Sets padding for a specified output character.

The argument *decimal-number* is the ASCII decimal representation of the character.

The argument *count* is the number of NUL bytes sent after that character.

The **no padding** subcommand removes padding for the specified output character.

[no] password *text*

Specifies a password.

The *text* argument may contain any alphanumeric characters, including spaces, up to 80 characters. The password checking is also case-sensitive.

The password *Secret* is different than the password *secret*, for example, and the password *two words* is an acceptable password.

The **no password** line subcommand removes the password.

[no] refuse-message *d message d*

Allows you to define the error message generated when a user attempts to connect to a line that is busy.

The argument *d* is a delimiting character of your choice.

The argument *message* is the message you want to show on the terminal.

Use the **no refuse-message** option do disable this command.

[no] rotary *group*

Define each group of lines. It adds a line to the specified rotary group.

The argument *group* is an integer you choose between 1 and 100 that identifies the rotary group.

[no] session-limit *session-number*

Sets the maximum number of concurrent sessions allowed. The argument *session-number* specifies the maximum number of sessions.

[no] session-timeout *minutes*

Sets the interval the protocol translator waits for input or output tyraffic before closing the connection to a remote computer and returning the terminal to the idle state.

The argument *minutes* specifies the interval in minutes.

The default interval is 0 (zero), indicating the protocol translator maintains the connection indefinitely.

[no] terminal-type *terminal-name*

The **terminal-type** subcommand records, in the argument *terminal-name*, the type of terminal connected to the line for use in Telnet terminal-type negotiation; the Telnet terminal-type negotiation uses *terminal-name* to inform the remote host of the terminal type. The text *terminal-name* is used by TN3270 for display management.

The **no terminal-type** subcommand removes the information about the type of the terminal.

transport input [telnet] [lat] [pad] [rlogin] [none]

Specifies which protocols may be used for input on a line.

transport output [telnet] [lat] [pad] [rlogin] [none]

Specifies which protocols may be used for output on a line.

transport preferred [telnet|lat|pad|rlogin|none]

Specifies the preferred protocol to use when a command does not specify one.

[no] vacant-message
vacant-message *d message d*

The **vacant-message** subcommand enables the banner to be displayed on the screen of an idle terminal.

The **vacant-message** subcommand without any arguments causes the default message to be displayed.

If a banner is desired, follow the **vacant-message** subcommand with one or more blank spaces and a delimiting character (*d*) you choose. Then type one or more lines of text (*message*), terminating the text with the second occurrence of the delimiting character.

width *columns*

Sets the number of characters (*columns*) on a single line of the attached terminal.

Interface Configuration Subcommand Summary

This section lists all of the configuration subcommands in alphabetical order:

encapsulation *encapsulation-type*

Specifies an encapsulation method for the interface.

For Ethernet and IEEE 802.3 interfaces, the argument *encapsulation-type* is **arpa**. The **arpa** keyword establishes ARPA Ethernet 2.0 encapsulation, which is the default encapsulation method for Ethernet interfaces.

For serial interfaces, the *encapsulation-type* argument identifies one of the following serial encapsulation types that Cisco Systems' software supports:

- **bfex25**—Blacker Front End Encryption X.25 operation
- **ddnx25-dce**—DDN X.25 DCE operation
- **ddnx25**—DDN X.25 DTE operation
- **frame-relay**—Frame Relay
- **hdh**—HDH Protocol
- **hdlc**—Cisco Systems HDLC Protocol
- **lapb-dce**—X.25 LAPB DCE operation
- **lapb**—X.25 LAPB DTE operation
- **multi-lapb-dce**—X.25 LAPB multiprotocol DCE operation
- **multi-lapb**—X.25 LAPB multiprotocol DTE operation
- **ppp**—Point-to-Point Protocol (PPP)
- **smds**—SMDS service
- **x25-dce**—X.25 DCE operation
- **x25**—X.25 DTE operation

error-threshold *milliseconds*

Specifies the frequency with which the error count will be reset, use the **error-threshold** interface subcommand.

The argument *milliseconds* is a time measurement in milliseconds. The default is 1000.

interface *type unit*

Identifies a specific interface for configuration and starts interface configuration command collection.

The argument *type* is **ethernet** for the Ethernet interface and **serial** for supported serial interfaces.

The argument *unit* is a positive integer that specifies the nth interface of type *type*,

starting with 0 for the first interface of that type. The protocol translator can have only one network interface of each type, so *unit* is always 0.

[no] shutdown

Prevents the transmission of packets on the specified interface by disabling the interface.

Use the **no shutdown** command to restart a disabled interface.

System Configuration EXEC Command Summary

This section lists system configuration EXEC commands in alphabetic order.

configure

Begins the configuration process.

disable

Disables privileged level command.

enable

Enables the privileged command level.