Chapter 1
System Configuration

1

This chapter describes how to configure the system. You will find information about the following tasks in this chapter:

- Setting global system characteristics such as the host name, console banner message, system buffer sizes and boot file specifications.
- Storing and booting system software images.
- Establishing system and line passwords and system security.
- Defining network services such as the IP Finger protocol.
- Enabling and directing the logging of system debugging messages.
- Configuring the console and virtual terminal lines.

The chapter "Startup and Basic Configuration" provides the procedures for basic system configuration using the **setup** facility, and for entering system configuration mode. The chapter "Terminal Server User Commands" provides an overview of system operation and user-level commands available with the terminal server.

A command summary is included at the end of the chapter.

Configuring the Global System Parameters

The following sections contain procedures and command descriptions for configuring the global system characteristics, host name and passwords, and configuring system security and system management functions. The global configuration commands described in the following sections are entered in configuration mode. See the section "The Configure Command" in the "Startup and Basic Configuration" chapter for the procedures to enter into this mode.

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 the **hostname** global configuration command.

The command has this syntax:

hostname *name*

The argument *name* is the new host name for the network server and is case-sensitive. The default host name is *TS* for all Cisco terminal servers.

Example:

This command changes the host name to *sandbox*:

```
hostname sandbox
```

Displaying 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 protocol translation, this command applies only to two-step translations.

The general form of the banner command is:

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

The **motd**, **exec**, and **incoming** keywords control when the banner message is displayed. The use of these keywords is discussed in the following paragraphs.

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

Follow the **banner** command with one or more blank spaces, type the delimiting character, and then one or more lines of the message, terminating the message with the second occurrence of the delimiting character. There is no limit to the amount of characters that can be used for the banner, with the exception of buffer limits and what is appropriate for a banner.

The order of display at startup is banner MOTD, login and password prompts, then EXEC banner.

Example:

The following example uses the # character 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 message-of-the-day (MOTD) banner, use the **banner motd** global configuration command. The command syntax follows:

banner motd *d message d*

This message-of-the-day type banner is displayed to all terminals connected and is useful for sending messages that effect all users, impending system shutdowns, for example.

The **banner** command without any keywords specified defaults to the **banner motd** command. When a new **banner motd** command is added to the configuration, it overwrites the existing **banner** command (with no keyword specified). Similarly, if a **banner** command is added to the configuration, any existing **banner motd** command is similarly overwritten.

Displaying a Banner with an EXEC Process

To be able to display a message on terminals with an interactive EXEC, use the **banner exec** global configuration command. The command syntax follows:

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).

Displaying an Incoming Message Banner

To display messages on terminals connected to reverse Telnet lines, use the **banner incoming** global configuration command. An *incoming* connection is one initiated from the Ethernet side of the protocol translator.

The command syntax follows:

banner incoming *d text d*

This command is useful for displaying messages and instructions directed to users of reverse Telnet connections. Reverse Telnet connections are described in more detail in "Configuring Reverse Connections" in this chapter.

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:

Consider a server used to access a modem pool for either dial-in or dial-out usage. Suppose you want to show a different message to the user depending on whether they are dialing in, or dialing out. Suppose further you want the message to explain that the server is going to be reloaded with new software. The following example shows how to use the **banner** global configuration command and the **no exec-banner** line subcommand to accomplish this setting.

```
! lines 1 through 15 are connected to modems.
line 1 15
modem inout
!
! Both messages are inappropriate for the VTYs.
line vty 0 4
no exec-banner
!
banner exec /
This is cisco Systems training group terminal server.

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 terminal server 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 needed. Other buffers are permanently allocated and cannot be destroyed. The **buffers** command allows a network administrator to adjust initial buffer pool settings and 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:

buffers {**small**|**middle**|**big**|**large**|**huge**} **{permanent|max-free|min-free|initial}** *number*
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, and so forth. 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 de-allocated 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.

The argument *number* specifies the number of buffers to be allocated.

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

Dynamic Buffer Sizing

An optional global configuration command for adjusting huge buffer settings is the **buffers huge size** command. As with the above command, use only after consulting with Cisco staff:

buffers huge size *number*
no buffers huge size *number*

The **buffers huge size** command dynamically resizes all huge buffers to the value that you supply. The buffer size cannot be lowered below the default. The **no** version of the command with the keyword and argument 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
```

In this example, the system will resize huge buffers to 20000 bytes:

```
buffers huge size 20000
```

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 chapter "System Management."

Displaying a "Host-Failed" Message

To display a specific message when a connection fails with a specified host, use the **busy-message** configuration command as follows:

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

The **busy-message** command defines a message that the terminal server displays whenever an attempt to connect to the specified host fails.

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.

This command applies only to Telnet connections. Defining a busy message for a host prevents all terminal server-initiated user messages, including the initial message that indicates the connection is "Trying..."

The **busy-message** command can be used in the **autocommand** to suppress these messages. See the section "Creating an Autocommand" in this chapter for more information.

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 host named *dross* fails:

```
busy-message dross #
Cannot 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 translation, this command is only useful in two-step translations. The command syntax follows:

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

This command applies only to rlogin and Telnet sessions. The **login-string** command defines a string of characters that the terminal server 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. To insert pauses into the login string, embed a percent sign (%) followed by the number of seconds to pause and the letter "p."

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.

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** configuration command:

lockable
no lockable

The **lockable** 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

A state machine allows packet dispatch based upon a sequence of characters, instead of a single character. The **dispatch-character** commands (described in the "Specifying a State Machine" section in this chapter) enable packets to be buffered, then transmitted upon receipt of a character. The **state-machine** commands allow 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 I C," for example).

State machines allow control of TCP 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 terminal server 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 terminal server takes based upon the assigned configuration commands, and the type of information received.)

The terminal server 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 the 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** command as follows:

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, go 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 zero. Characters that recur, and that have not been explicitly defined to have a particular action, return the state machine to state zero.

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

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 "Specifying a State Machine." The following two examples show two different ways to set up state machines:

Example 1:

This 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-timeout 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 insures that the characters from the function keys on an ANSI terminal are all lumped together into a single packet. This ensures that 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
```

Specifying Unusual Baud Rates

The Cisco terminal server derives the clock for 38400 bps from a programmable timer. The terminal server software lets the user re-program the timer to provide other baud rates that would otherwise be unavailable. The current software provides the ability to change this rate on a per-system basis only. To define extra baud rate for the box, use the **extra-baudrate** configuration command:

extra-baudrate *number*

The argument *number* defines the extra baudrate available for the box. The possible values are: 57600, 38400, 28800, 23040, 16457, 14400, 12800, and 11520. The value specified in an **extra-baudrate** command replaces the value of 38400 as a possible argument in the **terminal speed** EXEC command, described with "Changing the Terminal Baud Rate" in the chapter "Terminal Server User Commands."

After using this command, the individual lines can have their speed changed using the normal baud rate configuration commands (**tsxpeed**, **rxspeed**, **speed**).

Note: Using the **extra-baudrate** command immediately changes the speed of any lines that were using the old extra baudrate, but does not update the independent configuration (for example, the EXEC **show line** command will display a previously set rate of 14400, even if the port is currently running at 38400 baud). You should always reconfigure the speed of any lines using the **extra-baudrate** command after changing the rate.

Setting Configuration File Specifications

This section describes the **boot** global configuration commands used to configure boot files. The **boot** command can be used to perform these tasks:

- Change default file names.

- Specify a server host for netbooting configuration files and boot image files.
- Specify the size buffer to configure for netbooting a host or network configuration file.

The commands to load files over the network take effect the next time the software is reloaded, *provided they have been written into nonvolatile memory.* See the section "Storing the System Image in Flash Memory" for information about configuring and using the Flash memory (MC+) card.

Changing the Network Configuration File

The network configuration file contains commands that apply to all network servers on a network. The default name of this file is *network-config*. To change the name of the *network-config* file, use the **boot network** global configuration command. The full command syntax follows:

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 terminal server 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 full command syntax follows:

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

Specifying a Boot File Buffer Size

Normally, the terminal server uses a buffer the size of the system nonvolatile 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 follows:

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 nonvolatile 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.

Obtaining the System Image Over the Network

A terminal server can execute its system software stored in ROM, or it can be configured to load a different version across the network. Network loading is only supported on systems with at least four megabytes 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 nonvolatile memory option to specify netbooting, place a **boot system** command in nonvolatile 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.

Note: The file names of *flash* and *rom* are not allowed, as they are used to indicate that the Flash Memory, or system ROMs, respectively, are to be used for booting system images.

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 (see Figure 1-1).

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

Example:

This command uses the nonvolatile 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
```

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 terminal server scans this list until it successfully loads the appropriate network or host configuration file or system boot image. In this example, the terminal server 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 depend on the TFTP server you are loading the files from.

If the terminal server 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.

Storing the System Image in Flash Memory

Note: Use of the Flash EPROM is subject to the terms and conditions of the software license agreement that accompanies your Cisco product.

Following is an overview of the steps to configure the Flash card. This overview assumes that the CSC-MC+ card and the appropriate level of system software EPROMs are installed in your server. See the Cisco publication *Installing and Configuring the Flash Memory Card* (78-0889-01) for specific installation instructions.

- Set up your system to boot from ROM software. This requires setting the ROM-boot jumper on the processor card to pattern 0-0-0-1 (see Figure 1-1).

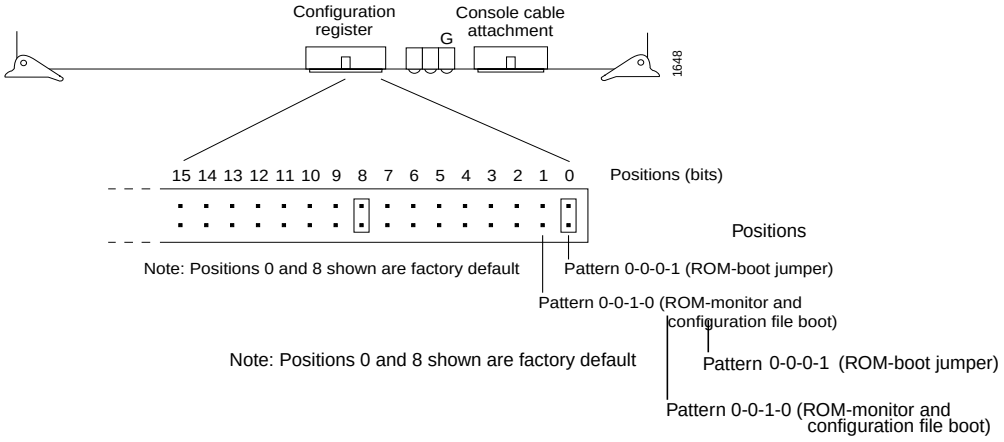
- Power the system on and allow it to boot up.
- Copy the TFTP image to Flash memory using the **copy tftp flash** command (see "Using the Flash Memory Card for Storing and Booting System Software" in the chapter "Startup and Basic Configuration" for more information about this command).
- Configure from the terminal to automatically boot from the desired file in Flash memory using the **boot system flash** configuration command; write your configuration to memory using the **write memory** command.
- Set your system to netboot from a desired file name using the **boot system** configuration command. This requires setting the jumper on the processor card to a pattern other than 0-0-0-0 or 0-0-0-1; pattern 0-0-1-0 is preferred for this (see Figure 1-1).

Note: Pattern 0-0-0-0 (ROM monitor) means that no auto-boot from ROM will be executed. Pattern 0-0-0-1 (ROM-boot) means that a ROM-boot will be executed. Pattern 0-0-1-0 (ROM monitor and configuration file boot) means that a configuration file boot will be executed. The pattern 0-0-0-1 is part of the factory default jumper positions for the processor card configuration register. The second default jumper is at position 8 (see Figure 1-1). Although this jumper is not used for the Flash function, it is required for normal operation and should not be removed at any time.

- Power-cycle and reboot your system to ensure that all is working as expected and that the configuration is stored in nonvolatile memory.

Once you have successfully installed and tested the CSC-MC+ card, you may want to configure the system with the **no boot system flash** command in order to revert back to booting from ROM.

Figure 1-1 CPU Configuration Register



Specifying ROM as a Source of the System Image

When netbooting, either with or without Flash memories, it is often desirable to be able to specify that the ROM image is to be booted if other boot images are not available.

boot system rom
no boot system rom

Use the **boot system rom** global configuration command to specify the use of the ROM system image when other **boot system** commands exist in the configuration.

Example:

This example illustrates a list specifying two possible internet locations for a system image, with the ROM software being used as a backup.

```
boot system ts3-rx.90-1 192.31.7.24
boot system ts3-rx.83-2 192.31.7.19
boot system rom
```

Note: The file names of *flash* and *rom* are not allowed, as they are used to indicate that the Flash Memory, or system ROMs, respectively, are to be used for booting system images.

Automatically Booting from Flash Memories

You can auto-boot from Flash using the **boot system flash** system configuration command:

boot system flash *filename*

Note: The **no boot system** configuration command disables all **boot system** configuration commands regardless of argument. Specifying the arguments **flash**, *filename*, or *IP address* with the **no boot system** command will disable only the command specified by these arguments.

The **boot system flash** command will boot the first valid file in Flash memory. The **boot system flash** *filename* will boot the file specified by this file name.

Configure the system to automatically boot from the desired file in Flash memory using the **configure terminal** command and the **boot system flash** *filename* system configuration command.

Note: If only one file is present in Flash memory, the file name argument is not necessary. The command **boot system flash** will boot that file.

The **no boot system flash** command disables the feature.

Write the configuration to nonvolatile memory with the **write memory** command. Following this, boot the system with the **reload** command. See the following example for an illustration of this procedure.

Example:

The following illustrates this sequence of commands:

```
TS#configure terminal
boot system flash tsxx
^Z
TS#write memory
[ok]
TS#reload
[confirm]<Return>
System Bootstrap, Version 4.5(0.3)
Copyright (c) 1986-1990 by cisco Systems
CSC3 processor with 4096 Kbytes of memory
F3: 1578668+35572+156084 at 0x1000
RRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRRR
RRRRRRRRRRRRRRRR.... [OK - 162132/2110400 bytes]
F3: 1586072+36028+156004 at 0x1000
{ROM Monitor copyrights...}
```

Configuring a Fault-Tolerant Booting Strategy

Occasionally network failures make booting across the network impossible. To lessen the effects of network failure, consider the following boot strategy.

After Flash is installed and configured, you can configure the system to boot in the following order:

1. Boot an image from Flash
2. Boot an image from a system server (netboot)
3. Boot from ROM image

This boot order will provide a more fault-tolerant alternative in the netbooting environment. The order of the commands needed to implement this strategy is shown in the following example:

```
TS#configure terminal
boot system flash tsxx
boot system tsxx 131.131.101.101
boot system rom
^Z
TS#write memory
```

```
[ok]
TS#
```

In addition to these commands, the ROM boot jumper on the configuration register of the processor card must be changed to pattern 0-0-0-1 (Position 0) see Figure 1-1.

Using this strategy, a system used primarily to boot across the network would have three alternative sources from which to boot. These alternative sources would help cushion the negative affects of a failure with the TFTP file server, and the network in general.

Establishing Passwords and System Security (TACACS)

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

You can set passwords to control access to the privileged command level and to individual lines. Additionally, you can configure a terminal server 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 terminal servers; Cisco patterned its TACACS support after the DDN application.

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

Establishing the Privileged-Level Password

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

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 does not prompt you for a password if the privileged mode password is not set. Additionally, if the enable password is not set and the line 0 (console line) password is not set, it is only possible to enter privileged mode on the console terminal. This feature allows you to use physical security rather than passwords to protect privileged mode if that is what you prefer to do.

If the enable password is not set and the line 0 (console) password is set, it is possible to enter privileged command mode either without entering a password at the console terminal or by entering the console line password when prompted while using any other line.

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 select the TACACS-style user ID and password checking mechanism instead, use the following subcommand:

login tacacs

To disable all password checking, use the command:

no login

The terminal server 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
```

Recovering from a Lost Password

If your server has the nonvolatile 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, force the server into factory diagnostic mode by turning off the server, inserting a jumper in bit 15 of the processor configuration register, and turning on the server. On the STS-10x, press the Break key while the system is first booting, and issue the "O/R" command at the ROM monitor. Next issue the "i" and "b" commands at the ROM monitor.

Follow these steps.

Step 1: You will be asked if you want to set the manufacturers' addresses. Respond by typing "Yes." You then see the following prompt:

```
TEST-SYSTEM>
```

Step 2: Type the **enable** command to get the privileged prompt:

```
TEST-SYSTEM#enable
```

Step 3: Type the **show configuration** command to review the system configuration and find the password. Do not change anything in the factory diagnostic mode.

Step 4: To resume normal operation, turn off the server, remove the jumper from bit 15 (or bit 7) of the configuration register, and turn on the server again.

Step 5: Log into the server with the password that was shown in the configuration file.

The processor configuration registers are described in the hardware configuration and reference publication for your particular product.

Note: All debugging capabilities are turned on during diagnostic mode.

Establishing Terminal Access Control

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

The configuration commands in the following paragraphs 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 follows:

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 follows:

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 follows:

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 five 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 follows:

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

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

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

Example:

This command changes the interval timer to ten 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 follows:

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 terminal server 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: Use the **tacacs-server-last-resort** subcommand to be sure that login can occur. For example, when the systems administrator needs to log into troubleshoot TACACS servers that might be down.

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

Establishing Privileged Level TACACS

Use the following variations of the **enable** command to configure privileged-level command access using the TACACS protocol.

Enabling the Privileged Mode

The **enable use-tacacs** global configuration command sets the TACACS protocol to determine whether a user can access the privileged command level. The command syntax follows:

enable use-tacacs

When you use this command, the EXEC **enable** command prompts for 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 Privileged Mode 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 follows:

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

The configuration commands in the following paragraphs 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 follows:

tacacs-server extended
no 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. Information includes responses from terminal servers 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 follows:

tacacs-server notify {connect|slip|enable|logout}

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

- **connect**—User makes connections
- **slip**—User turns SLIP on or off
- **enable**—User enters the **enable** mode
- **logout**—User logs out

Login Authentication

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

tacacs-server authenticate {connect|slip|enable}

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

- **connect**—User makes connections
- **slip**—SLIP connections
- **enable**—Use of **enable** command

Optional Password Verification

You can specify that the first TACACS request to a TACACS server be made *without* password verification. This option is configured with the **tacacs-server optional-passwords** global configuration command:

tacacs-server optional-passwords

When the user types in the login name, the login request is transmitted with the name and a zero-length password. If accepted, the login procedure completes. If the TACACS server refuses this request, the terminal server prompts for a password, and tries again when the user supplies a password. The TACACS server must support authentication for users without passwords to make use of this feature. This feature supports all TACACS requests—login, SLIP, enable, and so on.

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 terminal server software supports these needs by implementing a local **username** configuration command. The format for the command 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 **no password** 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 terminal server from disconnecting the user after an automatic command (set up with the **autocommand** keyword) has completed. Instead, the user gets another login prompt.

The **noescape** keyword prevents a user from using an escape character on the host to which he is connected.

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 terminal server, 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 might look like the following:

```
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 is as follows:

```
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 Systems' 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 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.

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 follows:

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

The server ignores packets from hosts that the access list denies. The access list 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 IP access list number.

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

Example:

This command allows the terminal server to process only those packets from hosts passing access list 21.

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

Setting the System Contact String

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

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 follows:

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 follows:

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 allowing 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 follows:

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 ten. 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 four 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 follows:

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 establishes a packet filtering of a maximum size of 1024 bytes:

```
snmp-server packetsize 1024
```

Establishing the TRAP Message Recipient

To specify the recipient of the TRAP message, use the **snmp-server host** global configuration command. The command syntax follows:

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

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 *community-string* is the password-like community string to send with the TRAP messages.

The optional keywords define the TRAPS are to be included, as follows:

- **snmp**—Causes all SNMP-type TRAP messages to be sent and sends the Cisco-specific RELOAD TRAP message
- **tty**—Causes TCP connection TRAP messages to be included

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

Example 1:

This example causes all the SNMP-type messages to be sent to the host specified by the name cisco.com. The command uses the community-string *comaccess* as a password. The command syntax follows:

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

Example 2:

By default, the terminal server sends all **tty** events to a host specified in the **snmp-server host** command. To disable the reporting of these events, you must explicitly specify so in a **no snmp-server host** command. To limit the sending of events to the seven SNMP traps, enter a global configuration command sequence as follows:

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

This sequence blocks sending of the **tty** events to the host at the specified address.

Establishing TRAP Message Authentication

To establish the TRAP message authentication, use the **snmp-server trap-authentication** global configuration command. The command syntax follows:

snmp-server trap-authentication
no snmp-server trap-authentication

This command 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. Note also that the community string is checked before any access list that may be set, so it is possible to get spurious TRAP messages. The only workarounds are to disable TRAP authentication or to configure an access list on a router between the SNMP agent and the SNMP manager to prevent packets from getting to the SNMP agent.

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 follows:

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 directly connected terminals, virtual terminals, and the terminal server'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:

snmp-server system-shutdown
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 using 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 terminal server after the terminal server has been started, use the **no snmp-server** global configuration command. The command syntax follows:

no snmp-server

Configuring the Trivial File Transfer Protocol (TFTP) Server

You can configure the terminal server to act as a limited Trivial File Transfer Protocol (TFTP) server from which other Cisco servers can boot their software. As a TFTP server host, the terminal server responds to TFTP read request messages by sending a copy of its ROM software to the requesting host. The TFTP read request message must use the file name that you specified in the network server configuration.

To specify TFTP server operation for a communications server, use the **tftp-server system** global configuration command. The full syntax follows.

tftp-server system *filename* **[***list***]**
no tftp-server system *filename* **[***list***]**

This command has two arguments: *filename* and *list*. The argument *filename* is the name you give the communications server ROM file, and the argument *list* is an IP access-list number.

The system sends a copy of the ROM software to any host which issues a TFTP read request with this file name. To learn how to specify an access list, see the "Configuring IP Access Lists" section in the chapter "TCP/IP Configuration and Management."

You can specify multiple file names by repeating the **tftp-server system** command. To remove a previously defined file name, use the **no tftp-server system** command and append the appropriate file name and an access-list number.

Example:

This command causes the terminal server to send, via TFTP, a copy of the ROM software when it receives a TFTP read request for the file *version-9.0*. The requesting host is checked against access list 22.

```
tftp-server system version-9.0 22
```

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 systems with nonvolatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; disabled by default, except on the STS-10x.
- **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.
- **tcp-keepalives-{in | out}**—Generates keepalive packets on idle network connections. The **in** keyword generates them on incoming connections (initiated by remote host); the **out** keyword generates them on outgoing connections (initiated by a user).

  The TCP keepalive capability allows a terminal server 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 terminal server 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 is a column in the EXEC **show tcp** display showing the keepalive statistics. The `wakeups` row shows how many keepalives have been transmitted without receiving any response (this is reset to zero when a response is received).
- **nagle**—Enables the Nagle algorithm for limiting TCP transactions. It effects only TCP traffic generated by the Cisco server. 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 for each TCP connection. This subcommand is useful in some networks with slow links. See RFC 896 for more information about use of the Nagle algorithm.

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

Examples:

To enable TFTP autoloading of configuration files, enter this subcommand:

```
service config
```

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

```
no service nagle
```

Redirecting System Error Messages

By default, the terminal server 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** global 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 terminal server.

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 "The Configure Command" in the chapter "Startup and Basic Configuration" for more details on this process.)

Enable the message logging using the **logging on** command. The command syntax follows:

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 follows:

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 follows:

logging console *level*
no logging console

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

The argument *level* is one of the keywords listed in Table 1-1, which describes each message priority level and lists the UNIX syslog definition.

Table 1-1 Error Message Logging Priorities

Level/Keyword	Description	Syslog Definition
0 emergencies	System unusable	LOG_EMERG
1 alerts	Immediate action needed	LOG_ALERT
2 critical	Critical conditions	LOG_CRIT
3 errors	Error conditions	LOG_ERR
4 warnings	Warning conditions	LOG_WARNING
5 notification	Normal but significant condition	LOG_NOTICE
6 informational	Informational messages only	LOG_INFO
7 debugging	Debugging messages	LOG_DEBUG

There are four categories of messages generated by the software:

- Error messages about software or hardware malfunctions at the LOG_ERR level.
- Output for the **debug** commands at the LOG_WARNING level.
- Interface up/down transitions and system restarts at the LOG_NOTICE level.
- Reload requests and low process stacks are at the LOG_INFO level.

The default is to log messages to the console at the warnings level (4). To display all levels of messages, set console logging to level 7 with the **debugging** keyword.

The EXEC command **show logging** displays the addresses and levels associated with the current logging setup, as well as any other logging statistics. The **no logging console** command disables logging to the console terminal.

Example:

By default, the console intitializes to the warning level. Any message with a priority less than or equal to level 4 will be printed on the console and any terminal line for which a **terminal monitor** EXEC command has been issued.

The following example changes the default to the alert level, so that messages with a priority less than or equal to 1 are displayed to the console.

```
logging console alerts
```

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 follows:

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 in Table 1-1.

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 a syslog server host, use the **logging** command with the appropriate Internet address. The command syntax follows:

logging *host*
no logging *host*

The **logging** command identifies a syslog server host to receive logging messages. By issuing this command more than once, you build a list of syslog servers that receive logging messages.

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

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 follows:

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 in this chapter.

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

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 *level* arguments list in Table 1-1 for other arguments that can be listed.)

The UNIX system sends messages at or above 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.

Note: Many UNIX systems require a tab character to be used as the "white space" separator in the /etc/syslog.conf file. Use of a space character rather than a tab may cause the entry in /etc/syslog.conf to be ignored.

Configuring Asynchronous Lines

This section describes the commands for configuring asynchronous lines and their characteristics.

To configure your 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 "The Configure Command" in the chapter "Startup and Basic Configuration" 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 follows:

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.
- **printer**—A parallel printer line.
- **tty**—A standard asynchronous line.
- **vty**—A virtual terminal for remote console access. The terminal server host can support 5 virtual terminals for access by incoming Telnet, LAT, or MOP connections.

Note that there are two ways to address lines: by absolute line number, or the line's position in the line table, and by relative line number, or by the type of line it is, either standard (tty) or virtual terminal line (vty). The line types are ranked as follows in the line table:

1. Console 0
2. Standard asynchronous line (tty)
3. Auxiliary port
4. Virtual terminal line (vty)

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 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 the absolute and relative line numbers, use the EXEC command **show users all.**

The terminal server displays an error message if you do not specify a line number.

Note: Line numbers are in octal on the ASM and MSM terminal servers, but they are in decimal form on the STS-10x terminal server.

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 asynchronous terminal lines, 0 through 4:

```
line tty 0 4
```

Configuring the CPU Auxiliary Port

The **line** command keyword **aux** allows use of an auxiliary RS-232 DTE port available on all processor cards. Use this port to attach to an RS-232 port of a CSU/DSU, protocol analyzer, or modem. You can monitor that port remotely 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 display of the EXEC command **show users all**), then the TCP port would be 2001.

Note: You cannot use the auxiliary port as a second console port, nor can you initiate connections from this port. Its purpose is to *receive* connections from remote systems.

When configuring the auxiliary port, address it using the **aux** keyword as line 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.

By default, the auxiliary port supports an EXEC process. This default can be re-enabled using the **exec** line subcommand.

You must order a special cable from Cisco Systems for use with this auxiliary port.

Example:

These commands configure the auxiliary port with a line speed of 2400 baud, and enable the EXEC.

```
line aux 0
exec
speed 2400
```

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 DTR signal will be active whenever an EXEC is configured on the auxiliary port.

Note: The EXEC is still present and may be used by the next person that dials into the number. This could cause security problems.

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 up to 80 alphanumeric characters, including spaces. 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 words "Speak Easy" as the password on virtual terminal line 4:

```
line vty 4
login
password Speak Easy
```

Turning On/Off the Vacant Banner

The terminal server displays a message on a 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 command syntaxes 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.

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

Example:

The following example turns on the system banner and displays this message:

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

Setting a "Lines in Use" Message

The terminal server allows you to specify the message to be displayed when an incoming connection is attempted and all rotary group or other lines are in use. If you do not define a message, when all lines are in use the user will receive a system-generated default message. You can use this additional text to provide the user with further instructions. Use the **refuse-message** command to define this message. The syntax of this command is as follows:

refuse-message *d message d*
no refuse-message

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.

When you define a message using this command, the terminal server does the following:

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

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

Example:

In the following example, line 5 is configured with the **refuse-message** command, and the user is given information about what to try next.

```
line 5
refuse-message  /The dial-out modem is currently in use.

Please try again later, or use the slow speed modem available as
```

```
hostname "dialout-slow."
/
```

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** command. The command syntax follows:

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** command. The command syntax follows:

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 configuration command. The command syntax follows:

insecure
no insecure

The information presented is only used by the LAT software, which reports such connections as dialups to remote systems.

In the 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** subcommand. The command syntax follows:

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 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** subcommand. The command syntax follows:

width *columns*

This subcommand sets the number of character *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: Some 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** subcommand. The command syntax follows:

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 Activation Character

To set the activation character, use the **activation-character** subcommand. The command syntax follows:

activation-character *decimal-number*
no activation-character

The **activation-character** subcommand defines the character you type at a vacant terminal to begin a terminal session.

The argument *decimal-number* is the ASCII decimal representation of the activation character. The default activation character is Return (ASCII character 13).

The **no activation-character** subcommand causes any character to activate a terminal.

Setting the Disconnect Character

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

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

This subcommand defines the character you type to end a session with the terminal server.

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

The Break character is represented by zero; NULL 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 Pause Screen Output Character

To define the local hold character used to pause output to the terminal screen, use the **hold-character** command. The command syntax follows:

hold-character *decimal-number*
no hold-character

The argument *decimal-number* is either the ASCII decimal representation of the hold character or a control sequence (for example, Ctrl-P). The Break character is represented by zero; NULL cannot be represented. By default, no local hold character is defined.

To continue the output, type any character after the hold character. To use the hold character in normal communications, precede it with the escape character.

Setting Terminal Parity

To define generation of a parity bit, use the **parity** subcommand. The command syntax follows:

parity {none|even|odd|space|mark}

The default is **none**.

Setting the Terminal Baud Rate

To set the terminal baud rate (speed of the lines), use one of the following **speed** subcommands. The command syntax follows:

speed *baud*
txspeed *baud*
rxspeed *baud*

The **speed** subcommand sets both the transmit (to terminal) and receive (from terminal) speeds.

The **txspeed** subcommand sets the transmit speed only.

The **rxspeed** subcommand sets the receive speed only.

The argument *baud* is typically set to 300, 1200, 2400, 4800, 9600, 19200, or 38400, but can be any user-defined speed in the range of 75 to 57600. The default speed is 9600 baud.

Setting the Data Bits

The **databits** subcommand sets the number of data bits per character, as follows:

databits {5|6|7|8}

If parity is being generated, specify 7 data bits per character. If no parity generation is in effect, specify 8 data bits per character. The default is 8 data bits per character.

Setting the Stop Bits

To set the number of the stop bits, use the **stopbits** command. The command syntax follows:

stopbits {1|1.5|2}

The **stopbits** subcommand sets the number of stop bits transmitted per byte. By default, the terminal server sends two stop bits.

Setting the Autobaud Detection

To set the line to autobaud, use the **autobaud** subcommand. The command syntax follows:

autobaud
no autobaud

The **autobaud** subcommand specifies automatic baud rate detection over a range from 300 to 19200 baud.

To start communications with the terminal server using autobaud detection, you type multiple Returns at your terminal. A 600, 1800, or 19200 baud line requires three Returns to detect the baud rate. A line at any other baud rate requires only two Returns. If you type extra Returns after the baud rate is detected, the EXEC simply displays another system prompt.

Note: A line set for autobaud cannot be used for outgoing connections. Nor can you set autobaud capability on a line using 19,200 baud when the parity bit is set because of hardware limitations.

Setting the Flow Control

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

flowcontrol {none|software [in|out]|hardware}

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

The keyword **none** specifies no flow control and is the default.

The keyword **software** sets software flow control. An additional keyword specifies the direction: i**n** causes the terminal server to listen to flow control from the attached device, and **out** causes the terminal server 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). You can change them with the **stop-character** and **start-character** subcommands described later in this section.

The keyword **hardware** sets hardware flow control. For information about hardware flow control, see the hardware reference manual for your server product. Only limited modem control is possible on a line using hardware flow control. Refer to "Configuring the Modem Control Lines" later in this chapter.

Setting the Flow Control Start Character

To set the flow control start character, use the **start-character** subcommand. The command syntax follows:

start-character *decimal-number*
no start-character

The **start-character** subcommand defines the character that signals the start of data transmission when software flow control is in effect. The argument *decimal-number* is the ASCII decimal representation of the start character. The default start character is Ctrl-Q (ASCII character 17). Use the **no start-character** command to remove the character.

Setting the Flow Control Stop Character

To set the flow control stop character, use the **stop-character** subcommand. The command syntax follows:

stop-character *decimal-number*
no stop-character

The **stop-character** subcommand defines the character that signals the end of data transmission when software flow control is in effect.

The argument *decimal-number* is the ASCII decimal representation of the stop character. The default stop character is Ctrl-S (ASCII character 19). Use the **no stop-character** command to remove the character.

Setting Character Padding

To set the padding on a specific output character, use the **padding** command. The command syntax follows:

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

The argument *decimal-number* is the ASCII decimal representation of the character. The argument *count* is the number of NULL bytes sent after that character. The **no padding** subcommand removes padding for the specified output character.

Example:

This example pads a Return (ASCII character 13) with 25 NULL bytes:

```
padding 13 25
```

Defining the Transport Protocol for a Specific Line

To determine the protocols allowed on individual lines, and to select default protocols, use the **transport** line configuration subcommands. There are three **transport** commands: **transport output**, **transport input**, and **transport preferred**. Selection of which command to use is determined by whether it is for incoming or outgoing connections, or as a method of preferred selection.

Specifying the Protocols Going Out on a Line

The **transport output** line subcommand specifies which protocols may be used for outgoing connections from a line, and has the following syntax:

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

By default, all protocols supported are allowed on the line.

Use one of these optional keywords to change the default:

- **telnet**—Selects the TCP/IP Telnet protocol. It allows a user at one site to establish a TCP connection to a login server at another site.
- **lat**—Selects the DEC LAT protocol, the Local Area Transport protocol used most often to connect terminal servers to DEC hosts.
- **rlogin**—Selects the UNIX rlogin protocol for TCP connections. 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.
- **pad**—Selects X.3 PAD, used most often to connect terminal servers to X.25 hosts.
- **none**—Prevents any protocol selection on the line. The system normally assumes that any unrecognized command is a hostname. If the protocol is set to **none**, the system no longer makes that assumption. No connection will be attempted if the command is not recognized.

An example using this command is included at the end of this section.

Specifying the Protocols to Connect to a Line

The **transport input** line subcommand allows the system administrator to define which protocols to use to connect *to* a specific line of the terminal server. It has the following syntax:

transport input [telnet|lat|pad|none]

This command can be useful in distributing resources among different types of users, or making certain that only specific hosts can access a particular port. When using protocol translation, the **transport input** command is also useful in controlling exactly which protocols can be translated to other protocols when using two-step translation.

By default, all protocols supported are allowed on the line. Use these optional keywords to change the default:

- **telnet**—Specifies all types of incoming TCP/IP connections.
- **lat**—Selects the DEC LAT protocol, and specifies both incoming reverse LAT and host-initiated connections.
- **pad**—Selects X.3 PAD incoming connections.
- **none**—Prevents any protocol selection on the line. This makes the port unusable by incoming connections.

Access lists for each individual protocol may be defined in addition to the allowances created by the **transport input** command.

An example using this command is included at the end of this section.

Specifying a Preferred Protocol

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 terminal servers that support LAT, the default protocol is LAT. For those that do not support LAT, the default is Telnet. The choices for this command follow:

- **telnet**—Selects the TCP/IP Telnet protocol. It allows a user at one site to establish a TCP connection to a login server at another site.
- **lat**—Selects the DEC LAT protocol, the Local Area Transport protocol used most often to connect terminal servers to DEC hosts.
- **rlogin**—Selects the UNIX rlogin protocol for TCP connections. 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.
- **pad**—Selects X.3 PAD used most often to connect terminal servers to X.25 hosts.

- **none**—Prevents any protocol selection on the line. The system normally assumes that any unrecognized command is a host name. If the protocol is set to **none**, the system no longer makes that assumption. No connection will be attempted if the command is not recognized.

Examples:

This example illustrates how to protect inbound connections.

```
! On a terminal server, don't allow any inbound connections to the ports
! that are actual terminals.  This prevents trojan horse programs from
! attaching to the port and deriving passwords.
!
line 1 20
location Undergrad terminal room
transport input none
```

This example sets specific lines for specific protocol translation.

```
! On a protocol translator, reserve approximately 1/3 of the available
lines
! for each possible transport protocol.  Also prevent users from "hop-
ping"
! off with the same protocol they came in with.
!
line vty 0 32
transport input lat
transport output telnet pad
!
line vty 33 65
transport input telnet
transport output pad lat
!
line vty 66 99
transport input pad
transport output lat telnet
```

This example sets the preferred protocol to rlogin on terminal line 1:

```
line tty 1
transport preferred rlogin
```

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

Creating an Autocommand

You can set it up so that a command that will automatically execute its function for you by using **autocommand**. The command syntax follows:

autocommand *command*

The **autocommand** causes whatever commands associated with it to be executed.

The argument *command* is any appropriate EXEC command, including the hostname and any switches that occur with the EXEC command.

Example:

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

```
autocommand connect dustbin uucp /stream
```

Setting the Dispatch Character and Character Buffer

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

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

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* command. The command syntax follows:

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

The argument *milliseconds* specifies the number of milliseconds the terminal server 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 timeout 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 terminal server to attempt to buffer characters into larger-sized packets for transmission to the remote host. The terminal server normally dispatches each character as it is typed.

Specifying a State Machine

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

dispatch-machine *name*

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

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 information about the state machine, see discussion on "Setting Up State Machines for TCP" in this chapter.

Turning the EXEC On or Off

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

exec
no exec

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

By default, the terminal server starts EXECs on all lines.

Suppressing EXEC Banners

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

exec-banner
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 asynchronous terminal lines 0 through 4:

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

Setting the EXEC Timeout Intervals

The EXEC command interpreter waits for a specified interval of time until the user starts 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 follows:

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 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 command sets a time interval of 2 minutes, 30 seconds:

```
exec-timeout 2 30
```

This command 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 input or output traffic, use the **session-timeout** command. The command syntax follows:

session-timeout *minutes* **[output]**
no session-timeout

This subcommand sets the interval that the terminal server waits for 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 optional keyword **output** specifies that when traffic is sent to an asynchronous line from the terminal server (within the specified interval), the connection is retained. If the keyword **output** is not specified, the session time-out interval is based solely on detected input from the user.

The default interval is zero, indicating the terminal server maintains the connection indefinitely.

Example:

The following subcommand sets an interval of 20 minutes, and specifies that the time-out is subject to traffic detected from the user (input only):

```
session-timeout 20
```

The following subcommand sets an interval of 10 minutes, subject to traffic on the line in either direction:

```
session-timeout 10 output
```

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 follows:

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

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

Saving User Changes Between Sessions

To save configurations between sessions, use the **private** subcommand. The command syntax is:

private
no private

The **private** subcommand ensures that the configuration options the user can set, will remain in effect between terminal sessions. This behavior is desirable for terminals in private offices.

By default, user-set configuration options are cleared with the EXEC command **exit** or when the interval set with the **exec-timeout** subcommand has passed.

Setting Input Notification

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

notify
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 chapter "Terminal Server User Commands."

The **no notify** subcommand ends notification.

Serial Line Configuration Examples

Example 1:

This 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), and with transmit and receive speeds set to 19200 baud. The **dispatch-timeout** subcommand causes the terminal server to wait 100 milliseconds after the first character is put into a packet before it is sent.

```
line 12
no exec
flowcontrol software
speed 19200
dispatch-timeout 100
<Ctrl-Z>
```

Example 2:

This example shows how to configure a printer line. The subcommands turn off the terminal EXEC, turn on software flow control, set the stop bits to 1, and set the line speed to the maximum (38400 baud) to handle the requirements of a laser printer.

```
line 13
location Laser Writer/pubs
no exec
flowcontrol software
stopbits 1
speed 38400
<Ctrl-Z>
```

Configuring the Modem Control Lines

Cisco Systems terminal servers use six RS-232 signals for each port, an arrangement that allows eight connections to be handled by one 50-pin Telco, RJ-11, or RJ-45 connector. The terminal server can support the most popular forms of modem control and/or hardware flow control as well as high-speed dial-up modems.

The RS-232 output signals are called TXDATA and DTR. The input signals are called RXDATA, CTS, and RING. The sixth signal is ground. Depending on the type of modem control, these names may or may not correspond to the standard RS-232 signals which have similar names.

You configure the different types of modem control using subcommands of the **line** configuration command.

This section describes the line subcommands for configuring modem control. State diagrams accompany some of the subcommand descriptions to illustrate how the modem control works.

The diagrams show two processes:

- The "create daemon" process creates a tty daemon that handles the incoming network connection.
- The "create EXEC" process creates the process that interprets user commands. (Refer to Figures 4-2 to 4-7.)

In the diagrams, the current signal state and the signal the line is watching are listed inside each box. The state of the line (as displayed by the EXEC command show line) is listed next to the box. Events that change that state appear in italic along the event path, with actions the software takes described within the ovals.

See the "Starting Line Configuration" section earlier in this manual for information about entering the **line** configuration command and its subcommands. Because of the way the terminal server software scans modem lines, the system operates more efficiently if all lines configured for modem control are contiguous.

To configure a modem control line, use the **modem** subcommand. The command syntax follows:

modem *keyword*
no modem *keyword*

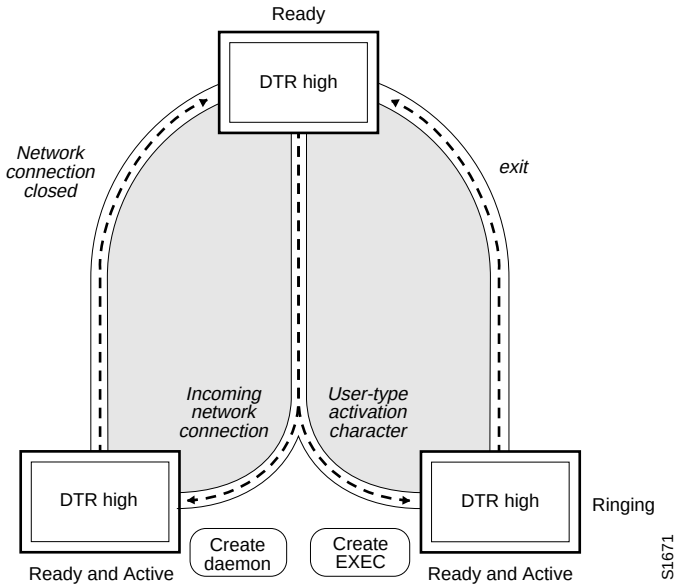
The argument *keyword* is one of the modem control keywords described in the following sections. For example, to configure a line for modem callout control, use the **modem callout** subcommand.

The **no modem** *keyword* subcommand removes modem control from a line.

By default, a terminal line is configured for no modem control.

Figure 1-2 illustrates line behavior when no modem control is set. The DTR output is always high, and CTS and RING are completely ignored. The terminal server creates an EXEC when the user types the activation character. Incoming TCP connections occur instantly if the line is not in use and can be closed only by the remote host.

Figure 1-2 EXEC and Daemon Creation on a Line with No Modem Control



Configuring DTR to Reflect Connection Status

To configure a line for low DTR, use the **modem dtr-active** subcommand. The command syntax follows:

modem dtr-active

The **modem dtr-active** subcommand configures a line to leave DTR low unless the line has an active incoming connection or EXEC.

This behavior can be useful if the line is connected to an external device (for example, a time-sharing system) that needs to know whether a line is in active use.

The **modem dtr-active** subcommand is similar to the **no modem** subcommand described earlier in this section.

Configuring a Line to Require CTS

To configure a line to require CTS, use the **modem cts-required** subcommand. The command syntax follows:

modem cts-required

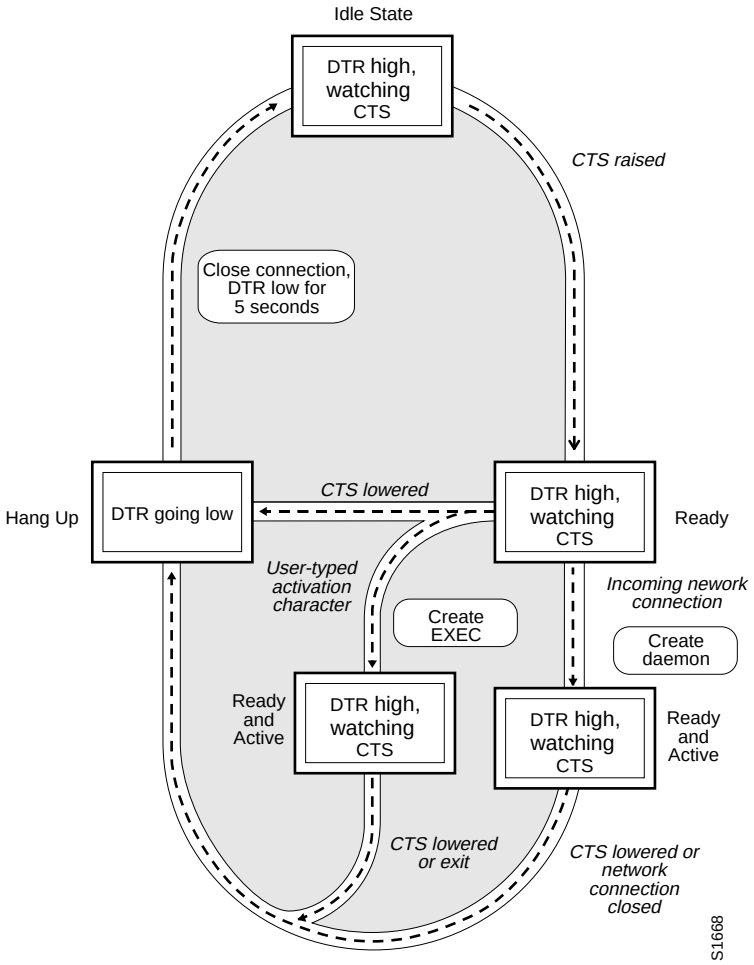
The **modem cts-required** subcommand supports lines that either the user or the network can activate.

This subcommand is useful for closing connections from a user's terminal when the terminal is turned off, and for preventing disabled printers and other devices in a *rotary group* from being considered. For more information about rotary groups, refer to the section on "Connections to One or More Lines (Rotary Group)" later in this chapter.

Figure 1-3 illustrates the **modem cts-required** process. This form of modem control requires that CTS be high throughout the use of the line. If CTS is not high, the user's typed input is ignored and incoming connections are refused (or step to the next line in a rotary group).

Note: In order for a terminal server to reliably detect a CTS signal change, the signal must remain in the new state for at least one full second.

Figure 1-3 EXEC and Daemon Creation on a Line Configured for Continuous CTS



Configuring a Line for a High-Speed Dial-Up Modem

To configure a line for a high-speed modem, use the **modem ri-is-cd** subcommand. The command syntax follows:

modem ri-is-cd

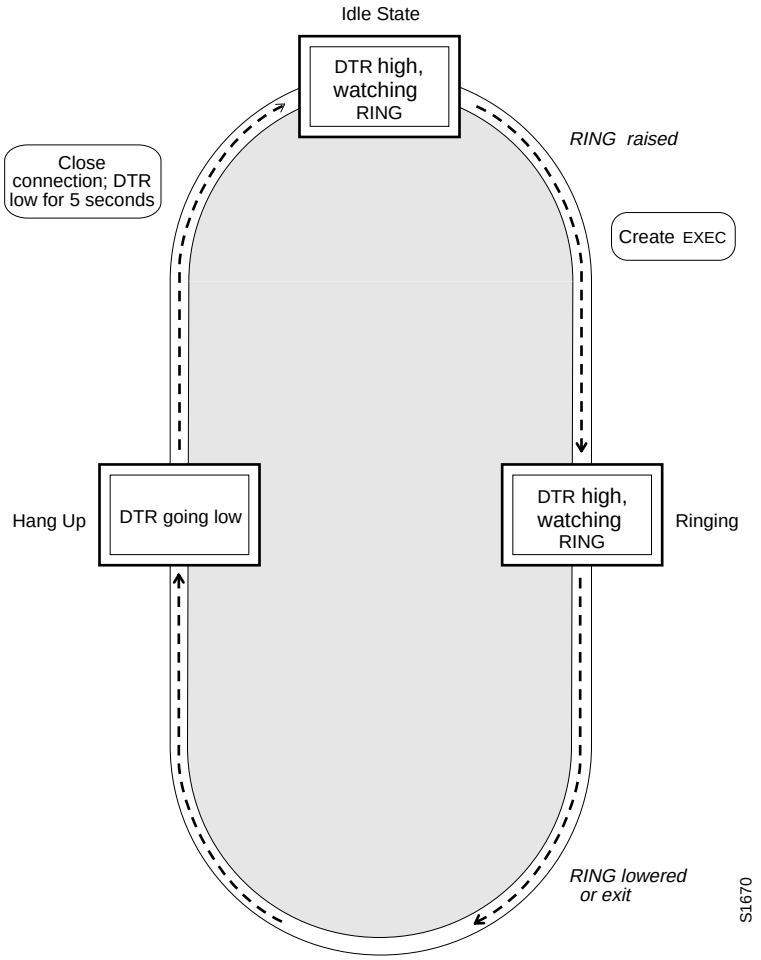
The **modem ri-is-cd** subcommand supports modems that can automatically handle telephone line activity, such as answering the telephone after a certain number of rings.

Configure the modem to answer the telephone on its own as long as DTR is high, drop connections when DTR is low, and use its CD signal to accurately reflect the presence of carrier. Wire the modem's CD signal (generally pin-8) to the terminal server's RING input (pin-22).

You can turn on the modem's hardware flow control independently to act on the status of the terminal server's CTS input. Wire CTS to whatever signal the modem uses for hardware flow control. If the modem expects to do hardware flow control in both directions, you may also need to wire the modem's flow control input to some other signal that the terminal server always has high (such as DTR).

Figure 1-4 illustrates the **modem ri-is-cd** process. When the terminal server detects a signal on the RING input of an idle line, it starts an EXEC or autobaud process on that line. If the RING signal disappears on an active line, the terminal server closes any open network connections and terminates the EXEC. If the user exits the EXEC or the terminal server terminates it because of no user input, the line hangs up the modem by lowering the DTR signal for five seconds. The modem is ready to accept another call after five seconds.

Figure 1-4 EXEC Creation on a Line Configured for a High-Speed Dial-up Modem



Configuring a Line for a Dial-In Modem

To configure a line for a dial-in modem, use the **modem callin** subcommand. The command syntax follows:

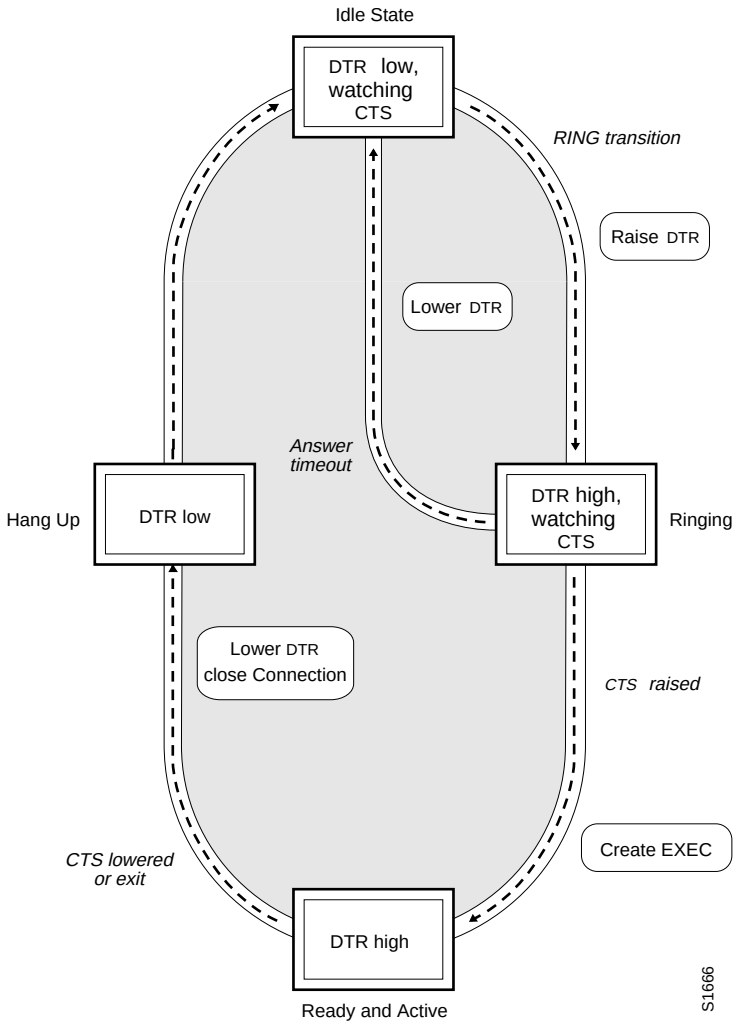
modem callin

The **modem callin** subcommand supports dial-in modems that use DTR to control the off-hook status of the telephone line. Raising the DTR signal answers the telephone; lowering DTR hangs up the telephone.

Figure 1-5 illustrates the **modem callin** process. When a modem callin line is idle, it has DTR in a low state and waits for a transition to occur on the RING input. This transition causes the line to raise DTR and start watching the CTS signal from the modem. After the modem raises CTS, the terminal server creates an EXEC on the line. If the time-out interval (set with the **modem answer-timeout** subcommand) passes before the modem raises CTS, the line lowers DTR and returns to the idle state.

Note: The **modem callin** and **modem cts-required** line subcommands are useful for SLIP operation. These subcommands ensure that, when the line is hung up or CTS drops, the line reverts from SLIP mode to normal interactive mode. These subcommands do not work if you use the **slip dedicated** subcommand to put the line in SLIP mode permanently.

Figure 1-5 EXEC Creation on a Line Configured for Modem Callin



Although you can use the **modem callin** subcommand with newer modems, the **modem ri-is-cd** subcommand described earlier in this section is more appropriate. The **modem ri-is-cd** command frees up CTS for hardware flow control. Modern modems do not require the assertion of DTR to take a phone line off-hook.

Configuring a Line for Reverse Connections

To configure a line for reverse connections, use the **modem callout** subcommand. The command syntax follows:

modem callout

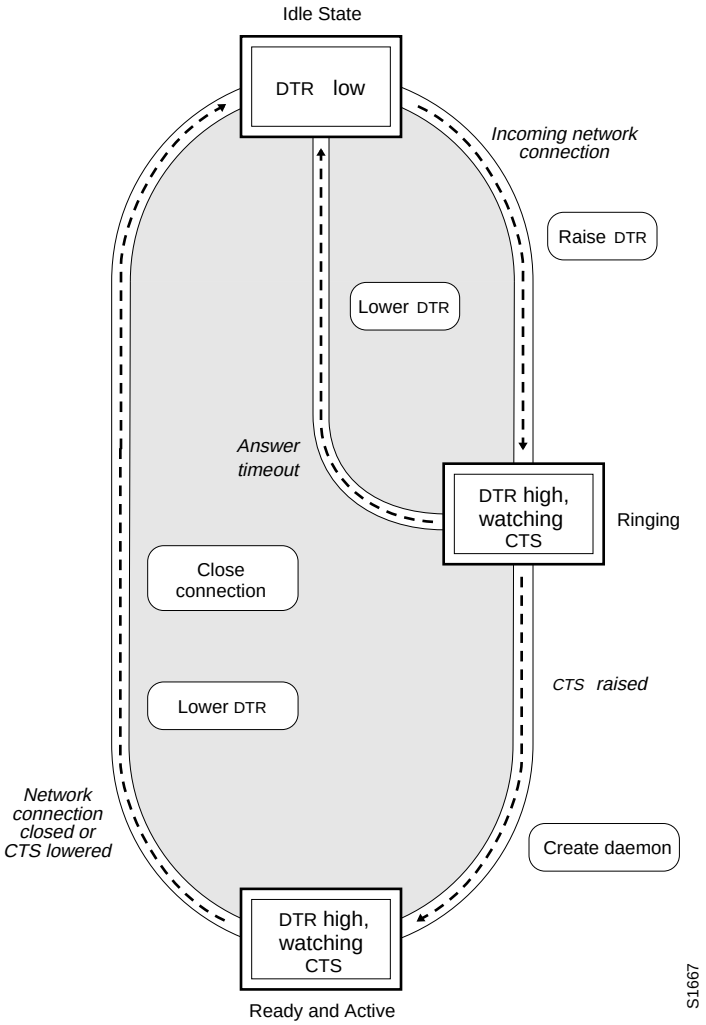
The **modem callout** subcommand supports ports connected to computers that expect to be connected to modems, an arrangement often called “milking machine” mode. (For more information, refer to the section on “Configuring Reverse Connections” later in this chapter.) This subcommand causes the terminal server to behave somewhat like a modem.

Figure 1-6 illustrates the **modem callout** process. When the terminal server receives an incoming connection, it raises DTR and waits to see if the CTS becomes high as an indication that the host has noticed its signal. If the host does not respond within the interval set with the **modem answer-timeout** subcommand, the terminal server lowers DTR and drops the connection.

Note: The line configuration subcommand **modem callout** is typically used to prevent incoming calls.

Modem callout lines used in a rotary group can take a long time to cycle through inactive lines (15 seconds is the default for each line) before connecting to a responsive device or finding no responsive device and dropping the network connection.

Figure 1-6 Daemon Creation on a Line Configured for Modem Callout



Configuring a Line for Both Incoming and Outgoing Calls

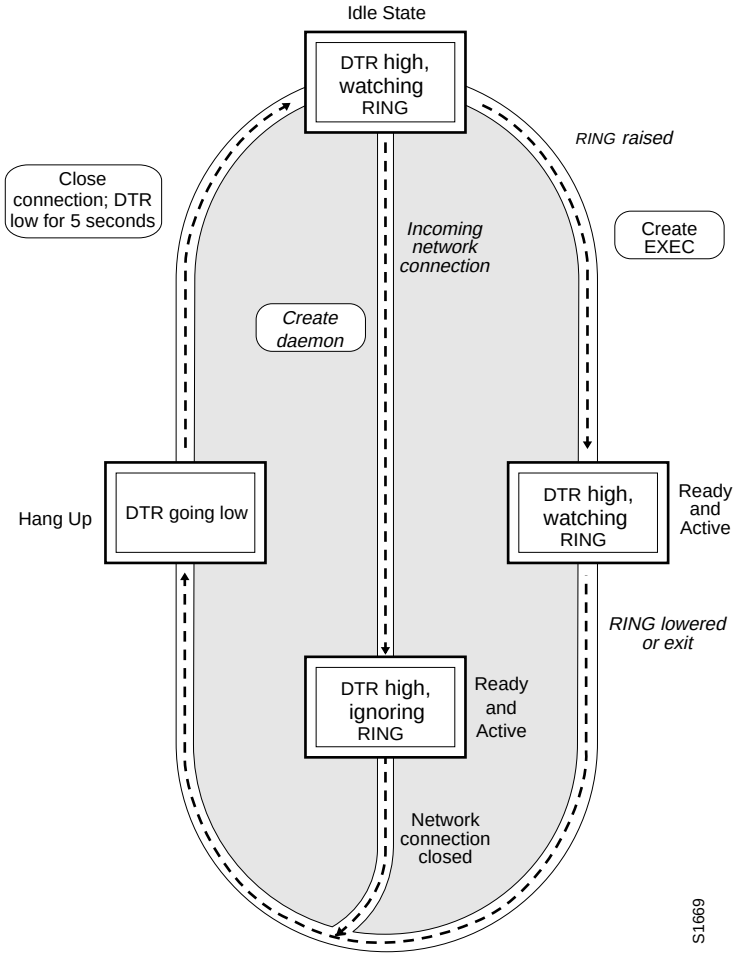
To configure a line for both incoming and outgoing calls, use the **modem inout** subcommand. The command syntax follows:

modem inout

The **modem inout** subcommand enables a line to be used for both incoming and outgoing calls on dial-in/dial-out modems. Note that the terminal server does not support any dialing protocols. Therefore, the host system software or the user must provide any special dialing commands when using the modem for outgoing calls.

Figure 1-7 illustrates the **modem inout** process. If the line is activated by raising RING, it behaves exactly as a line configured with the **modem ri-is-cd** subcommand, described earlier. If the line is activated by an incoming TCP connection, the line behaves similarly to a nonmodem line.

Figure 1-7 EXEC and Daemon Creation on a Line Configured for Incoming and Outgoing Calls



Note: If your system incorporates dial-out modems, consider using access lists to prevent unauthorized use.

Configuring Line Modem Timing

To configure line modem timing, use the **modem answer-timeout** subcommand. The command syntax follows:

modem answer-timeout *seconds*

The **modem answer-timeout** subcommand sets the interval during which the terminal server raises DTR in response to RING and the modem responds to CTS. This behavior is useful for modems that take a long time to synchronize to the appropriate line speed.

The argument *seconds* specifies the interval in seconds.

The default is 15 seconds.

Configuring Automatic Line Disconnect

To configure automatic line disconnect, use the **autohangup** subcommand. The command syntax follows:

autohangup

The **autohangup** 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.

Using High-Speed Modems

Dial-up modems that operate over normal dial-up telephone lines at speeds of 9600 baud and higher are now available. These modems do not operate at a guaranteed throughput; instead, they operate at a speed dependent on the quality of the line, the effectiveness of data compression algorithms on the data being transmitted, and other variables. These modems use hardware flow control to stop the data from the host (by toggling some RS-232 signal) when they cannot take any more.

In addition to hardware flow control, dial-up modems require special software handling. For example, they must be configured to create an EXEC when a user dials in and to hang up when the user exits the EXEC. These modems must also be configured to close any existing network connections if the telephone line hangs up in the middle of a session.

Cisco Systems terminal servers support hardware flow control on their CTS input, which is also used by the normal modem handshake. To support both modem control and hardware flow control on the same line, use the **modem ri-is-cd** subcommand described earlier in this section.

The following commands may also be useful as you configure and use a high-speed modem:

- The **flowcontrol hardware** line subcommand enables outgoing hardware flow control based on the CTS input.
- The privileged EXEC command **debug modem** displays informational messages about modem control events, such as signal transitions and autobaud progress, on the console terminal. (These messages are also sent to any line that has issued a **terminal monitor** command; refer to the section on "Displaying Debug Messages on the Console and Terminals" in the chapter "Terminal Server User Commands.")
- The EXEC command **show line** displays the status of a line. In the detailed command output, a *Status* line with "Idle" identifies inactive modem ri-is-cd lines and all other modem lines; a *Status* line with "Ready" identifies lines in use.
- The privileged EXEC command **clear line** closes all the connections on a line and hangs up the modem. For more information, refer to "Clearing a Line" in the chapter "System Management."

Configuring Reverse Connections

In addition to initiating connections, the terminal server can receive incoming connections. This capability enables you to attach serial and parallel printers, modems, and other shared peripherals to the terminal server and drive them remotely from other systems.

The specific TCP port or socket to which you attach the device determines the type of service the terminal server provides on that line. When you attach the serial lines of a computer system or a data terminal switch to the serial lines of the terminal server, the terminal server acts as a network front-end for a host that does not support the TCP/IP protocols. This arrangement is sometimes called front-ending, reverse connection mode, or *milking machine* mode.

Note: Comparable information for DEC LAT is covered in the chapter "LAT Configuration and Management."

Connections to an Individual Line

Connections to an individual line are most useful when a dial-out modem, parallel printer, or serial printer is attached to that terminal server line. To connect to an individual line, the remote host or terminal must specify a particular TCP port on the terminal server. If Telnet protocols are required, that port is 2000 (decimal) plus the decimal value of the line number.

Note: Line numbers are octal on the ASM and MSM terminal servers and decimal on the STS-10x terminal server. On the ASM, for example, line 40 corresponds to decimal 32; the corresponding TCP port is 2032. On the STS-10x, the corresponding TCP port for line 9 would be 2009. There is no octal-to-decimal translation. The **service decimal-tty** global configuration command controls the specification of octal or decimal line numbers.

If a raw TCP stream is required, the port is 4000 (decimal) plus the decimal line number. The raw TCP stream is usually the required mode for sending data to a printer.

The Telnet protocol requires that carriage return characters be translated into carriage returns and line-feed character pairs. You can turn this translation off by specifying the Telnet binary mode option. To specify this option, connect to port 6000 (decimal) plus the decimal line number.

Example:

Assume that a laser printer is attached to line 10 of an MSM terminal server. Such a printer usually uses XON/XOFF software flow control. Because the terminal server will not receive an incoming connection if the line already has a process, you must ensure that an EXEC is not accidentally started. You must therefore configure it as follows:

```
line 10
flowcontrol software
no exec
```

A host that wants to send data to the printer would connect to the terminal server on TCP port 4008, send the data, and then close the connection. (Remember that line number 10 octal equals 8 decimal.)

Connections to One or More Lines (Rotary Group)

Connections may be made to the next free line in a group of lines, also called a rotary or hunt group. A line may be in only one rotary group; 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.

The Line Subcommand for Rotary Groups

You define each group of lines with the **rotary** subcommand of the **line** configuration command. The command syntax follows:

rotary *group*
no rotary

The **rotary** subcommand adds a line to the specified rotary group. The argument *group* is an integer between 1 and 100 that you choose to identify the rotary group. To list the defined rotary groups, use the privileged EXEC command **show line**. For more information, see "Maintaining the Asynchronous Lines" in the chapter "System Management."

The remote host must specify a particular TCP port on the terminal server to connect to a rotary group with connections to an individual line. 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

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 identifier 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 5005.

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

Rotary Connection Features

Connections to a rotary group can also take advantage of these features:

- Clear to Send—If a line in a rotary group is configured to require CTS, the terminal server skips that line if CTS from the attached device is low. This feature enables the terminal server to automatically avoid inactive host ports. To enable this feature, use the **modem cts-required** subcommand of the **line** configuration command. See the discussion on "Configuring the Modem Control Lines" earlier in this section.
- RS-232 Handshaking—Rotary groups are often associated with large terminal switches that require an RS-232 handshake before forming a connection. In this case, use the **modem callout** line subcommand to configure the lines in the group. If the RS-232 handshake fails on a line, the terminal server steps to the next free line in the rotary group and restarts the negotiation.
- Access Control—Use access lists for rotary groups as described in "Establishing Passwords and System Security (TACACS)" to restrict incoming connections on rotary groups.

- Session Timeout—Use the **session-timeout** subcommand of the **line** configuration command to set an interval for a line so that if no activity occurs on a remotely initiated connection for that interval the terminal server closes the connection. The terminal server assumes the host has crashed or is otherwise inaccessible.

Terminal Server Network Configuration Examples

This section presents sample network and host configuration files for a terminal server. You can place configuration commands in either or both file types. See "The Configure Command" in the chapter "Startup and Basic Configuration" for information about the **configure** command and the **interface** and **line** configuration commands. The last line in a configuration file must be the **end** keyword.

Duplicating configuration information between the network and the host configuration files does not present a problem. Furthermore, you can place all configuration commands in the host configuration file, reducing the network configuration file to just the **end** keyword.

However, if conflicting configuration information exists in the network and the host configuration files, the information in the host file takes precedence. This fact allows you to configure all network hosts in general and adjust for specific hosts as necessary.

Comment lines are entered by including an exclamation point (!) followed by a space and some text, or by simply pressing the Return key to leave a blank line and improve readability.

Network Configuration File

The network configuration file contains configuration parameters common to all Cisco Systems terminal servers, routers, and protocol translators on a network. The following list shows a sample network configuration file.

```
! The "network-confg" file for our network
!
! Define the names and addresses of some special hosts.
ip host chaff 192.31.7.18
ip host fuzz 192.31.7.45
!
! Use a domain server to look up host names.
service domain
no service ipname
name-server 192.31.7.19
domain-name CISCO.COM
!
end
```

Host Configuration File

The host configuration file contains configuration parameters specific to a terminal server or router host. The following list shows a sample host configuration file that is more complex than most. Generally, a host configuration file contains only a small subset of these commands.

Use the command **no exec** for unused terminals; setting the speed to a very low number (that is, 110) may prevent loading under certain conditions:

```
! This is a sample configuration file for chaff.
!
! Define the interface address and subnetting.
interface ethernet 0
ip address 192.31.7.18  255.255.255.0
!
! Define the string to be used in forming the EXEC prompt.
hostname chaff
!
! Define a password for the privileged mode.
enable password secret
!
! Define a default gateway only if the network does not have a gateway
! that supports proxy ARP.
ip default-gateway 192.31.7.67
!
! Access list 1 restricts access to hosts on net 192.31.7.0.
access-list 1 permit 192.31.7.0  0.0.0.255
access-list 1 deny 0.0.0.0 255.255.255.255
!
! Set a password on line 0, the console.
line 0
location ASM Console
password alsosecret
!
! Lines 1 through 7 are in a public terminal area. They are restricted
! to making connections to hosts on the local network. They also run at
! 4800 baud and have padding on the Return character.
line 1 7
location Public Terminal Pool
access-class 1 out
speed 4800
pad 13 25
!
! Terminal lines 10 through 16 are dial-in modems that
! can run from 300 to 19,200 baud.
line 10 16
modem callin
autobaud
!
! Lines 17 through 32 are hooked up in reverse connection mode to
! a large terminal switch. By Telnetting to TCP port 3001 on this
! host, a user gets the next free line in the rotary group.
line 17 32
rotary 1
modem callout
!
! Line 33 has a laser printer attached. A network printer daemon
! connects to TCP port 4027 on chaff to send data to the printer.
! Software flow control is used and the line speed is set to 38,400 baud.
line 33
location Laser Printer/pubs
no exec
flowcontrol software in
flowcontrol software out
```

```
stopbits 1
txspeed 38400
rxspeed 38400
!
! Lines 34 through 40 are unused at present. Their line
! speeds are set to zero.
line 34 40
location Unused Terminals
speed 0
!
! There are 5 virtual terminals. The vty keyword makes their exact
! line numbers unnecessary. Only hosts specified by
! access list 1 may connect to the virtual terminals.
line vty 0 4
access-class 1 in
!
! The end keyword terminates a command file.
end
```

Global Configuration Command Summary

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

banner [exec|incoming|motd] *d message d*

Modifies the banner name for the terminal server. Follow the **banner** command with one or more blank spaces, a delimiting character (*d*), then type one or more lines of text (*message*), terminating the message with the second occurrence of the delimiting character.

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

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

The **motd** keyword displays a message-of-the-day type banner whenever a line is activated or when an incoming Telnet connection is created.

[no] boot buffersize *bytes*

Specifies the size of the buffer to be used for netbooting a host or a network configuration file. The argument *bytes* specifies the size of the buffer to be used. By default, it is the size of nonvolatile memory, and there is no minimum or maximum size that may be specified.

[no] boot {host|network|system} *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.

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 terminal server 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.

The keyword **system** indicates that the file name and host addresses for booting operating software over the network are in nonvolatile memory. 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] boot system flash *filename*

Auto-boots system from the Flash memories. Specify a name to associate with the image from Flash memories with the *filename* parameter.

[no] boot system rom

Auto boot the system from the ROM system image. This command is usually used as a backup to other **boot system** commands which specify system images that exist either on the network or in Flash memories.

[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 terminal server 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 a delimiting character (*d*), then type one or more lines of text (*message*), terminating the message with the second occurrence of the delimiting character.

[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 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.**

enable use-tacacs

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

extra-baudrate *number*

Defines extra baud rates for the system, in place of 38400. The argument *number* defines the extra baud rates available for the system. Possible baud rate values are: 57600, 38400, 28800, 23040, 16457, 14400, 12800, and 11520.

hostname *name*

Modifies the host name for the terminal server. 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. 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. 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 above *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 listed in the **logging console** command description. 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 listed in the **logging console** command description. The **no logging trap** command disables logging to syslog servers.

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

Defines a string of characters that the terminal server 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*), then type one or more lines of text *(message*), terminating the message with the second occurrence of the delimiting character. The options effect display of the message:

- **%***sec***p**—Sets a pause in seconds.
- **%***sec***w**—Prevents users from issuing commands or keystrokes during a pause.
- **%b**—Sends a Break character.

This command applies only to rlogin and Telnet sessions.

no snmp-server

Disables the SNMP server after it has been started.

[no] service *keyword*

Tailors network-based services. The argument *keyword* is one of the following:

- **config**—Specifies TFTP autoloading of configuration files; disabled by default on systems with nonvolatile memory.
- **decimal-tty**—Specifies that line numbers be displayed and interpreted as decimal numbers rather than octal numbers; disabled by default, except on the STS-10x.
- **finger**—Allows Finger protocol requests (defined in RFC 742).
- **tcp-keepalives-{in|out}**—Generates keepalive packets on idle network connections. The **in** keyword generates them on incoming connections; the **out** keyword generates them on outgoing connections.
- **nagle**—Enables the Nagle algorithm for limiting TCP transactions. Disabled by default.

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

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

Sets up an access list that determines which hosts can send requests to the terminal server and sends all traps to the host. The argument *list* is an integer from 1 through 99 that specifies an access list. The access list applies only to the global read-only SNMP agent configured with the command **snmp-server community**. The **no snmp-server access-list** command removes the specified access list.

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

Sets up the community access string and enables SNMP server operation on the terminal 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** 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* **[snmp|tty]**

Specifies which hosts should receive TRAP messages. 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 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. 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

This command requests a "shutdown-after-message" and 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. 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, go to the specified next state. Full 8-bit character comparisons are done, so the maximum value is 255. Make sure 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.

The optional **transmit** keyword causes the packet to be transmitted, and the state machine to be reset to state 0. Characters 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 milliseconds, and the state reset to zero.

[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: Keyword **connect** specifies that the user makes TCP connections. Keyword **slip** specifies SLIP connections. Keyword **enable** specifies 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 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.

[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 password** configuration 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.

tacacs-server notify {connect|slip|enable|logout}

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

The optional keywords are used to specify notification of the TACACS server whenever users do one of the following:

- **connect**—Make TCP connections
- **slip**—Turn SLIP on or off (terminal server only)
- **enable**–Enter the **enable** command
- **logout**—Log out

tacacs-server optional-passwords

Specifies that the first TACACS request to a TACACS server be made *without* password verification. When the user types in the login name, the login request is transmitted with the name and a zero-length password. If accepted, the login procedure completes. If the TACACS server refuses this request, the terminal server prompts for a password, and tries again when the user supplies a password. The TACACS server must support authentication for users without passwords to make use of this feature. This feature supports all TACACS requests—login, SLIP, enable, and so on.

[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.

[no] tftp-server system *filename list*

Specifies TFTP server operation for a communications server. The argument *filename* is the name given to the communications server ROM file, and the argument *list* is an IP access-list number. The system sends a copy of the ROM software to any host which issues a TFTP read request with this file name. You can specify multiple file names by repeating the **tftp-server system** command. To remove a previously defined file name, use the **no tftp-server system** command and append the appropriate file name and an access-list number.

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

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 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 **username** 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 terminal server 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.

The **noescape** keyword prevents a user from using an escape character on the host to which he is connected.

Line Configuration Subcommand Summary

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

[no] activation-character *decimal-number*

Sets the activation character. The **activation-character** subcommand defines the character you type at a vacant terminal to begin a terminal session. The argument *decimal-number* is the ASCII decimal representation of the activation character. The default activation character is Return (ASCII character 13). The **no activation-character** subcommand causes any character to activate a terminal.

[no] autobaud

Sets the line to autobaud. The **autobaud** subcommand specifies automatic baud-rate detection over a range from 300 to 19200 baud. The **no autobaud** subcommand disables the autobaud feature.

autocommand *command*

Automatically executes the EXEC command defined by the *command* argument. The command can include any argument normally associated with it such as host names or switch.

autohangup

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

databits {5|6|7|8}

Sets the number of data bits per character. If parity is being generated, specify 7 data bits per character. If no parity generation is in effect, specify 8 data bits per character. The default is 8 data bits per character.

[no] disconnect-character *decimal-number*

Defines the character typed to end a session with the terminal server. 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.

[no] dispatch-timeout *milliseconds*

Sets the dispatch timer. The argument *milliseconds* specifies the number of milliseconds the terminal server waits after putting the first character into a packet buffer before sending the packet. During this interval, more characters can 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. 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 ^ X. The **no escape-character** subcommand reinstates the default.

[no] exec

This EXEC subcommand determines whether or not the terminal server will start an EXEC process on the line. By default, the terminal server starts EXECs on all lines.

[no] exec-banner

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.

[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 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]|hardware}

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

The keyword **software** sets software flow control. An additional keyword specifies the direction: **in** causes the terminal server to listen to flow control from the attached device, and **out** causes the terminal server to send flow control information to the attached device. If not specified, both directions are assumed.

The keyword **hardware** sets hardware flow control. No modem control is necessary on a line utilizing hardware flow control.

By default, no flow control is set for a line. Use the **none** keyword to re-instate the default.

[no] hold-character *decimal-number*

Sets, changes, or removes the local hold character used to pause output to the terminal screen. The argument *decimal-number* is either the ASCII decimal representation of the hold character or a control sequence (for example, Ctrl-C). The Break character is represented by zero; NULL cannot be represented. By default, no local hold character is defined.

[no] insecure

Sets the line as in an insecure location. This line subcommand applies only to LAT connections.

length *screen-length*

Sets the terminal screen length. The argument *screen-length* is the number of lines on the screen. Default is 24 lines.

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, and is one of the following:

- **console**—Primary command line for the terminal server.
- **aux**—Enables use of an auxiliary RS-232 DTE port available on all processor cards. (This port is not available on the STS-10x.)
- **printer**—A parallel printer line.
- **tty**—A standard asynchronous line.
- **vty**—Virtual terminal for remote console access. The terminal server host can support five virtual terminals for access by incoming Telnet, LAT, or MOP 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 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*, then *first-line* and *last-line* are absolute rather than relative line numbers.

[no] location *text*

Enters a description about the terminal location and/or status into the configuration file. The argument *text* is the desired description.

[no] login

Causes the terminal server 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 password checking and allows 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.

modem answer-timeout *seconds*

Sets the interval during which the terminal server raises DTR in response to RING and the modem responds to CTS. This behavior is useful for modems that take a long time to synchronize to the appropriate line speed. The argument *seconds* specifies the interval in seconds. The default is 15 seconds.

modem callin

Configures support for dial-in modems that use DTR to control the off-hook status of the telephone line. Raising the DTR signal answers the telephone; lowering DTR hangs up the telephone.

modem callout

Configures support of ports connected to computers that expect to be connected to modems, an arrangement often called *milking machine* mode. This subcommand causes the terminal server to behave somewhat like a modem.

modem cts-required

Configures modem control lines to support lines that either the user or the network can activate.

modem dtr-active

Configures a line to leave DTR low unless the line has an active TCP connection in either direction.

modem inout

Enables a line to be used for both incoming and outgoing calls on dial-in/dial-out modems. Note that the terminal server does not support any dialing protocols. Therefore, the host system software or the user must provide any special dialing commands when using the modem for outgoing calls.

modem ri-is-cd

Configures support for modems that can handle telephone line protocols, such as answering the telephone after a certain number of rings.

no modem

Removes modem control from a line. By default, a terminal line is configured for no modem control.

[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. 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 NULL bytes sent after that character. The **no padding** subcommand removes padding for the specified output character.

parity {none|even|odd|space|mark}

Defines the generation of the parity bit. By default, the terminal server does no parity checking.

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

[no] private

Ensures that the configuration options the user can set remain in effect between terminal sessions. This behavior is desirable for terminals in private offices. By default, user-set configuration options are cleared with the EXEC command exit or when the interval set with the **exec-timeout** subcommand has passed.

[no] refuse-message *d message d*

Defines an error message that is displayed when a user attempts to connect to a line that is busy. The argument *d* is a delimiting character which denotes the beginning and end of the text entered for the *message* argument.

[no] rotary *group*

Defines a group of lines as a 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* **[output]**

Sets the interval the terminal server waits for traffic 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 optional keyword **output** specifies that when traffic is sent to an asynchronous line from the terminal server (within the specified interval), the connection is retained. If not specified, the session time-out interval is based only on user input, not output. The default interval is zero, indicating the terminal server maintains the connection indefinitely.

speed *baud*
txspeed *baud*
rxspeed *baud*

The **speed** subcommand sets both the transmit (to terminal) and receive (from terminal) speeds.

The **txspeed** subcommand sets the transmit speed only.

The **rxspeed** subcommand sets the receive speed only.

The argument *baud* can be 300, 1200, 2400, 4800, 9600, 19200, or 38400. The default speed is 9600 baud.

[no] start-character *decimal-number*

Sets or removes the character that signals the start of data transmission when software flow control is in effect. The argument *decimal-number* is the ASCII decimal representation of the start character. The default start character is Ctrl-Q (ASCII character 17).

stopbits {1|1.5|2}

Sets the number of stop bits transmitted per byte. By default, the terminal server sends 2 stop bits.

[no] stop-character *decimal-number*

Sets or removes the character that signals the end of data transmission when software flow control is in effect. The argument *decimal-number* is the ASCII decimal representation of the stop character. The default stop character is Ctrl-S (ASCII character 19).

[no] terminal-type *terminal-name*

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. Terminal type is used by TN3270 for display management. The **no terminal-type** subcommand removes the information about the terminal type.

transport input [telnet|lat|pad|none]

Specifies which protocols may be used for connecting *to* a line. By default, all protocols supported are allowed. Use one of the keywords to change the default.

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

Specifies which protocols may be used for outgoing connections from a line.By default, all protocols supported are allowed. Use one of the keywords to change the default.

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

Specifies the preferred protocol to use for outgoing connections when a command does not specify one.For terminal servers that support LAT, the default protocol is LAT. For those that do not support LAT, the default is Telnet.

[no] vacant-message [*d message d***]**

The **vacant-message** subcommand displays a banner 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 *columns* on an attached terminal. Default is 80 columns.

System Configuration EXEC Command Summary

This section lists the system configuration EXEC commands in alphabetic order.

configure

Begins configuration.

copy flash tftp

Copies a Flash TFTP image back to a TFTP server.

copy tftp flash

Copies a TFTP image into the current Flash configuration.

disable

Disables the privileged command level.

enable

Enables the privileged command level.