Chapter 3
Using the Protocol Translator

3

This chapter provides an introduction to using Cisco protocol translators. The emphasis here is on the various elements of a Cisco protocol translator's user interface. The following topics are addressed:

- Basic discussion of connection service features
- An overview of one-step and two-step translation processes
- Running the system and accessing different operational modes
- Procedures for making rudimentary connections
- An outline of common user connection service tools available with all protocols supported by Cisco's protocol translators

---

Note: This chapter is a survey of Cisco's protocol translator connection capabilities. The objective is to provide users with the essentials for making basic connections. Other chapters provide more depth in terms of configuration and customization.

Please refer to the chapter, "First-Time Startup," for specifics concerning system configuration using the **setup** command facility. Refer to the chapter, "Configuring the System," for details about system-wide configuration features. See the chapter "Configuring Translation," about the **translate** command documented in this section. the chapter, "Managing the System," addresses facilities for monitoring and managing Cisco protocol translators. Configuration utilities associated with specific transmission protocols (TCP/IP, DEC LAT, IBM TN3270, X.25, and XRemote) are addressed in the"Transmission Protocols." section of this guide. If you have special applications, or wish to customize your connections, please refer to these other chapters for details.

Using the EXEC Command Interpreter

Before exploring the specific connection commands provided with Cisco protocol translators, you should take time to familiarize yourself with the basic command line user interface. This section introduces you to the basic Cisco protocol translator user interface environment. This environment is generally referred to as the EXEC Command Interpreter (or EXEC).

The EXEC interprets the commands you type and carries out the corresponding operations. You can type commands when you see the system prompt, which is the host name followed by an angle bracket (>).

Command Use

You may type commands in uppercase, lowercase, or both uppercase and lowercase letters. You may also abbreviate commands and other keywords to the number of characters that cause the command to be a unique abbreviation. For example, you can abbreviate the **show** command to **sh**.

If you make a typing mistake, you can erase characters with the Delete or the Backspace key. Press either key to erase the last character you typed. To erase the entire line, type Ctrl-U. (This notation means "Hold down the Ctrl key and press the U key.")

The protocol translator acts on most commands after you press the Return key. You can abort a command at any time by typing Ctrl-C.

Note: All standard Cisco Ctrl-key conventions are described in the chapter "Configuring the System."

To list available EXEC commands, type a question mark (**?**). You can often enter a question mark (help command) to obtain more information about commands. For example, type `terminal ?` to obtain a list of **terminal** commands or `show ?` to obtain a list of **show** commands.

Certain EXEC commands produce multiple screens of output. At the end of each screen, the EXEC pauses and displays:

```
–More–
```

Type a space to continue the output; type anything else to return to the system command prompt.

Command Levels

For security purposes, the EXEC has two levels of access: *user* and *privileged*. The commands available at the user level are a subset of the commands available at the privileged level. Because many of the privileged commands set protocol translator operating parameters, the privileged level should be password-protected to prevent its unauthorized use. The system prompt for the privileged level ends with a pound sign (#) instead of an angle bracket (>).

The **enable** command allows access to the privileged level, prompting for a password if one has been set with the **enable password** configuration command. For more information, see the "Establishing Passwords and System Security" section in the chapter "Configuring the System," of this manual. To list available EXEC commands, use the **?** (question mark) command. At the user level, the **?** command produces this list:

```
cpt>?
connect <host>  Connect to host - same as typing just a host name
disconnect <cn> Break the connection specified by name or number
exit, quit      Exit from the EXEC
lat <service>   Connect to service using DEC LAT protocol
lock            Lock the terminal
login           Log in as a particular user
name-connection Give a connection a logical name
pad             Connect to host using PAD protocol
resume          Make the named connection be current
rlogin <host>   Connect to host using rlogin protocol
show <cmd>      Information commands, type "show ?" for list
systat          Show terminal lines and users
telnet <host>   Connect to host using telnet protocol
tn3270 <host>   Connect to host using telnet protocol (3270)
terminal        Change terminal's parameters, type "terminal ?"
where           Show open connections
xremote         Enter xremote mode
<cr>            To resume connection
```

At the privileged level, the **?** command lists the full protocol translator command set:

```
cpt#?
clear           Reinitialization functions, type "clear ?" for list
configure       Configure from terminal or over network
connect <host>  Connect to host - same as typing just a host name
debug           Enable debugging functions, type "debug ?" for list
disable         Turn off privileged commands
disconnect <cn> Break the connection specified by name or number
enable          Turn on privileged commands
exit, quit      Exit from the EXEC
lat <service>   Connect to service using DEC LAT protocol
lock            Lock the terminal
login           Log in as a particular user
name-connection Give a connection a logical name
pad             Connect to host using PAD protocol
ping            Send echo messages
reload          Halt and reload system
resume          Make the named connection be current
rlogin <host>   Connect to host using rlogin protocol
send <line>|*   Send message to a terminal line or lines
setup           Initialize system configuration
show <cmd>      Information commands, type "show ?" for list
systat          Show terminal lines and users
telnet <host>   Connect to host using telnet protocol
tn3270 <host>   Connect to host using telnet protocol (3270)
terminal        Change terminal's parameters, type "terminal ?"
test            Run hardware tests, type "test ?"
trace <address> Trace route to <address>
undebug         Disable debugging functions, type "undebug ?" for list
where           Show open connections
write           Write configuration memory, type "write ?" for list
xremote         Enter xremote mode
<cr>            To resume connection
```

Protocol Translation Methods

Protocol translation allows two systems running different networking protocols to communicate as if the protocols were the same. This can be a difficult task, as networking protocols may require different numbers of network connections to run an application, or may have different formats for their data and commands.

For this reason, there may never be a protocol translator that handles electronic mail or file transfer in a completely transparent manner. Fortunately, there are fewer differences among protocols that connect remote terminals to a computer (called virtual terminal protocols).

The protocol translator attempts to provide transparent protocol translation between systems running disparate protocols. Cisco protocol translation fully-supports two-way virtual terminal protocol translation between nodes running X.25, DEC LAT, and TCP/Telnet. Limited translation (two-step only) is supported for XRemote (to X.25 PAD environments) and TN3270 (to LAT, X.25, and TCP/Telnet). This allows terminal users on one network to access hosts (or host-like devices) on another network, despite differences in the native protocol stacks associated with the originating device and the target host. You can use either of two methods to accomplish this, as discussed in the following sections.

The One-Step Method

Note: One-step translation applies only to translations involving TCP (Telnet), LAT, and X.25. Protocol translators do not support one-step translations for TN3270 or XRemote.

The one-step method invokes a *translation session* process to provide fully transparent protocol conversion. In this method, the protocol translator masquerades as two or more hosts on the same network. When a connection is made to the protocol translator, it determines which host the connection is for, and what protocol that host is using. It then establishes a new network connection, using the networking protocol required by that host.

This has the advantage of being simpler from the users' point of view. It is more efficient, and allows the protocol translator to act upon greater knowledge of the protocols in use, as the protocol translator acts as a network connection rather than a terminal. One disadvantage, however, is that the initiating computer or user does not know that two networking protocols are being used. This creates the limitation that parameters of the foreign network protocols cannot be changed, once the connections are established.

The exception to this limitation is the set of parameters common to both networking protocols. Any parameter in this set can be changed, from the first host to the final destination.

To support connections in each direction, you must specifically include **translate** command statements in the configuration file to handle bi-directional connections.

Note: Refer tothe chapter, “Configuring Translation,” for more details concerning how to configure protocol translators for one-step translation using the **translate** command and for examples of how to build one-step connection environments.

An Example: One-Step Method (TCP to X.25 Host)

A typical one-step translation application might feature a UNIX workstation user attempting a connection to a remote X.25 host, named *host1*, over an X.25 PDN. In this case, the protocol translator automatically converts the Telnet connection request to an X.25 connection request and transmits the request as specified in the system configuration.

A connection is established by first entering the **telnet** command at the UNIX workstation system prompt, as follows:

```
unix%telnet host1
```

Note: This example implicitly assumes that the name *host1* is known to the UNIX host (obtained via DNS, IEN116, or a static table) and is mapped to the IP address used in a **translate** command. Refer to the chapter, “Translation Applications,” for examples of how to define and use the **translate** command.

The protocol translator accepts the Telnet connection, and immediately forms an outgoing connection with the remotely-located *host1* as specified in a **translate** command specified in your protocol translator’s active configuration file. Next, *host1* sets several X.3 parameters, including local echo. Since the Telnet connection is already set to local echo (at the UNIX host), no changes are made on the TCP connection.

The *host1* connection prompts for a user name, which is provided (and echoed locally), as follows:

```
login name:
```

Then *host1* sets the X.3 parameter to cause remote echo (the same process as setting X.3 PAD parameter 2:0), and prompts for a password. The protocol translator converts this to a Telnet option request on the UNIX host, which then stops the local echo mode.

At this point the user is connected to the PAD application and the application will set the X.3 PAD parameters (although they can always be overridden by the user). When the user is finished with the connection, he enters the escape character to exit back to the host connection, then enters the appropriate command to close the connection.

The *host1* host immediately closes the X.25 connection. The protocol translator then drops the TCP connection, leaving the user back at the UNIX system prompt.

The Two-Step Method

In the two-step method, a user explicitly establishes a network connection to a protocol translator, and from there establishes an outgoing network connection on a different type of network. The two steps are:

Step 1: Establish the incoming connection directly to the protocol translator.

Step 2: Establish the outgoing connection established from the protocol translator to another network host.

Note: In general, the two-step process is applicable when you want to use a protocol translator as a *general purpose gateway* between two types of networks (for example, X.25 PDN and TCP/IP). Instead of configuring every possible connection via embedded **translate** commands, the two-step method allows users greater flexibility in terms of connecting to network resources accessible via the protocol translator. The two-step process also allows another level of security, if TACACS global commands or the **login** line subcommand are configured. These security features are described in the chapter, "Configuring the System."

This process is similar to connecting a group of terminal lines from a PAD, to a group of terminal lines from a TCP terminal server. The difference is that you do not encounter the wiring complexity, unreliability, management problems, and performance bottlenecks that occur when two separate devices are connected via asynchronous serial lines. With the two-step connection process, a user can individually modify the parameters of either network connection, even while a session is in process.

The protocol translator supports the two-step model in both directions (for example, from Telnet to PAD, and the reverse).

Note: You must use the two-step method for translations of TN3270 and XRemote. Please refer to the chapters "Configuring TN3270" and "Configuring XRemote," for more information concerning these protocols.

Changing Parameters and Settings Dynamically

If you use the two-step method to connect, you can change PAD and local terminal parameters dynamically during a session.

The following example shows how you might make a dynamic change during a session. Suppose that you wanted to type in some information at the *Information Place*, and need to change the X.3 PAD parameters that define the editing characters from the default Delete key setting to the Ctrl-D sequence.

To do this you would first enter the escape sequence to return to the protocol translator system EXEC:

```
    Ctrl-^X
```

Then at the EXEC, type in the **resume** command (discussed in the chapter “Using the Protocol Translator”) with the **/set** keyword and the desired X.3 parameters.

```
    cpt>resume /set 16:4
```

You may also want to set the **/debug** switch to check that your parameter setting has not been changed by the host PAD. The **/debug** switch provides helpful information about the connection.

At the EXEC, you may also set a packet dispatch character or sequence using the **terminal dispatch-character** command. The **dispatch-character** line subcommand is discussed in the chapter “Configuring the System.” This example shows how to set ESC as a dispatch character using the EXEC command version of **dispatch-character**:

```
    cpt>terminal dispatch-character 27
```

To return to the PAD connection, simply enter:

```
    cpt>resume
```

Now you may use your new settings for your session.

An Example: The Two-Step Method (TCP to PAD)

The first step in any two-step translation is to connect directly from a terminal or workstation to a protocol translator. For example, a user might make the following connection request:

```
    unix_host%telnet cpt
```

If a protocol translator named *cpt* is reachable, it returns a login message and the user enters a user name and password. The next step then requires that the user connect from the protocol translator to some target system. For example, the user might attempt a connection to an X.25 host using the **pad** command. The user might enter the following:

```
    cpt>pad 71330
```

Once the connection is established, the protocol translator immediately sets the PAD to single character mode with local echoing, since this is the behavior the protocol translator expects. The PAD responds with its login messages and a prompt for a password:

```
    Trying 71330...Open
    Welcome to the Information Place
    Password:
```

As the password should not echo on the user’s terminal, the PAD requests remote echoing so that characters will be exchanged between the PAD and the protocol translator, but not echoed locally or displayed. After the password is verified, the PAD again requests local echoing from the protocol translator, which it does from then on.

Eventually, the user logs off, and is returned to the protocol translator system EXEC prompt. From there, the user executes the **quit** command and the protocol translator drops the network connection to the PAD.

Please refer to the chapter, “Configuring X.25 and X.3,” for details about the **pad** command.

Making Connections with a Cisco Protocol Translator

Users can make simple automated terminal-to-host connections, or can establish custom terminal sessions tailored to specific terminal or application requirements. The following connection descriptions define the basic connection capabilities available to users and the syntax associated with each user level EXEC connection command. Refer to the appropriate chapters of this manual, “Transmission Protocols,” for details about protocol-related customization of terminal characteristics. The commands outlined here represent the basic connection services provided by Cisco protocol translators for DEC LAT, Telnet, TN3270, XRemote, and X.25.

Note: In many instances, extensive optional argument strings can be concatenated to the basic commands. In general, these options, and supporting configuration capabilities, are reserved for discussion in the protocol-specific chapters of this manual.

You can change the default connection protocols discussed in this section by using the **transport** commands. For more information about the **transport** commands refer to the chapter, “Configuring the System.”

LAT Connections

DEC’s Local Area Transport (LAT) protocol is the protocol used most often to connect protocol translators to DEC hosts. LAT is a DEC-proprietary protocol. Cisco uses LAT technology licensed from DEC.

The LAT protocol is similar to TCP/IP’s Telnet protocol in that it allows a user at one site to establish a connection to a host at another site, then passes the keystrokes from one system to the other. A user can establish a LAT connection through the protocol translator to a DEC host, simply by entering the host name.

Establishing a LAT Connection

Use the EXEC command **lat** to establish a connection to a LAT learned service. The command has this syntax:

lat *name* **[node** *nodename* **| port** *portname* **| /debug]**

Enter this command at the EXEC prompt. It can be entered simply by typing the name of the service to which you want to attach, or with the optional keywords listed in the above syntax description. You can also enter the service name to make the connection.

The optional keyword **node** specifies connection to a specific LAT node which offers a service. Enter the name of the node in place of the *nodename* argument. If you do not include the node option, the node with the highest rating offering the service is used. Use the EXEC command **show lat nodes** to display information about all known LAT nodes (see the section "Monitoring LAT" in the chapter "Configuring LAT," for more information about this command).

The optional keyword **port** specifies a destination LAT port name. This keyword is ignored in most timesharing systems, but is used by terminal servers offering *reverse LAT* services. Enter the port name in the format of the remote system in place of the *portname* argument.

Reverse LAT services are discussed in the section "Inbound Session Support," in the chapter "Configuring LAT." Generally, *reverse LAT* involves connecting to a terminal server from a protocol translator. In this case, the target terminal server runs the host portion of the protocol.

The optional keyword **/debug** is a switch that, when enabled, prints parameter changes and other special messages on the terminal.

Note: With Cisco's implementation of LAT, you are not required to enter the word **lat** to establish a LAT connection. If you prefer, you can just enter the LAT learned service name. To show a listing of the available LAT learned services, use the **show lat services** command (see the section "Monitoring LAT" in the chapter "Configuring LAT" for more information about this command).

Sample Session:

The following example establishes a LAT connection from the protocol translator named *cpt* to the host *Eng2*.

```
cpt>lat eng2
Trying ENG2...Open
         ENG2 – VAX/VMS V5.2
Username:JSmith
Password:
    Welcome to VAX/VMS version V5.2 on node ENG2
    Last interactive login on Friday,  6-APR-1990 19:46
```

The LAT protocol is explicitly specified in this example. You specify the protocol when your preferred transport is set to "none" or to another protocol. The protocol translator responds with "Trying <system>..." and then "Open." If the connection were not successful you would receive a failure message. Refer to the chapter, "Configuring the System," for more information about setting terminal preferences.

Rlogin Connections

The **rlogin** command starts a connection using the UNIX rlogin protocol. The full syntax of the command follows:

rlogin *host* **[debug]**

The argument *host* is a host name or an Internet address. The optional keyword **debug** enables debugging output from the rlogin protocol.

An alternative to the Telnet protocol, the rlogin protocol provides superior flow control and output suppression. The Cisco implementation of rlogin does not subscribe to the "trusted" host model. That is, a user cannot automatically log onto a UNIX system from the protocol translator, but must provide a user ID and a password for each connection.

For more information concerning rlogin connections, please refer to the chapter, "Configuring TCP/IP," later in this manual.

Telnet Connections

This section addresses the basics of making Telnet connections. For more information concerning TCP/IP connections, please refer to the chapter, "Configuring TCP/IP," later in this manual.

Starting a Telnet Connection

Use the EXEC command **connect** or **telnet** to start a Telnet connection. The command has this syntax:

{connect | telnet} *host* **[***port***] [/***keyword***]**

The argument *host* is a host name or an Internet address. The optional argument *port* is a decimal TCP port number; the default is the Telnet server port (decimal 23) on the host. The optional argument *keyword* is one of the following:

- **/route:** *path*—Specifies loose source routing. The argument *path* is a list of host names or Internet addresses that specify network nodes, ending with the final destination. For example, the following command routes packets from the source system to *kl.sri.com*, then to 10.1.0.11, and finally to *mathom*:

  ```
  cpt>connect mathom /route:kl.sri.com 10.1.0.11 mathom
  ```

- **/line**—Enables Telnet line mode. In this mode, the protocol translator does not send any data to the host until the user presses Return. The user can edit the line using the standard protocol translator command editing characters (Backspace, Delete, Ctrl-U, Ctrl-W). The **/line** keyword is a local switch; the remote server is not notified of the mode change.
- **/debug**—Enables Telnet debugging mode.

- **/stream**—Turns on "stream" processing, which enables a raw TCP stream with no Telnet control sequences. A stream connection does no processing of Telnet options, and may be appropriate for connections to ports running UUCP and other non-Telnet protocols.

Note: With Cisco's implementation of TCP/IP, you are not required to enter the word **connect** or **telnet** to establish a Telnet connection. If you prefer, you can just enter the learned host name. To show a listing of the available hosts, use the **show hosts** command (see the section "Monitoring TCP/IP" in the chapter "Configuring TCP/IP," for more information about this command).

The protocol translator assigns a logical name to each connection; several commands use these names to identify connections. The logical name is the same as the host name, unless that name is already in use or you change the connection name with the EXEC command **name-connection**. If the name is already in use, the protocol translator assigns a null name to the connection.

TN3270 Connections

If your network administrator has configured an appropriate default terminal type, then using the EXEC command **tn3270** is all you need to enter to start a tn3270 session via a Cisco protocol translator. The syntax for this command is as follows:

tn3270 *hostname*

The argument *hostname* is the name of a specific host on a network that is reachable by the protocol translator. The default terminal emulation mode allows access using a VT100 emulation. If your terminal requires a different emulation, you must configure your protocol translator to support different terminal types.

The following example use of the EXEC command **tn3270** results in the Cisco protocol translator attempting to establish a terminal session with an IBM host named *finance.*

Example:

```
cpt>tn3270 finance
```

Note: Unlike Telnet and LAT connections, you *must* enter the command **tn3270** in order to make a connection to a host with this command.

Establishing connections using alternative configurations (which must be included in the Cisco protocol translator configuration) is explained in the next section. The process for configuring the Cisco protocol translator to include alternative (and custom) terminal emulations is outlined in the chapter, "Configuring TN3270," later in this manual.

XRemote Connections

If your host computer includes a server for XDMCP, such as the *xdm* program included in X11R4, you can use automatic session startup to initiate an XRemote session on an NCD X terminal.

The EXEC command used to initiate XRemote using XDMCP is **xremote xdm**. This command has the following syntax:

xremote xdm *hostname*

This command causes an XDMCP session start-up request to be made to the computer specified (*hostname*). If a *hostname* is not specified, a broadcast message is sent to all hosts. The first host to respond by starting up a session is used.

The protocol translator and X terminal stay in XRemote mode until either the display manager terminates the session, or a reset request is received from the X terminal.

Note: If your host does not include a server for XDMCP, you must perform the manual startup procedures outline in the chapter, "Configuring XRemote."

X.25 Transport Support

Cisco protocol translators support the use of X.25 as a transport mechanism for IP-based traffic. As you would expect, this capability covers Telnet, connect and rlogin connection commands, but also extends to XRemote and TN3270. Refer to the chapter, "Configuring X.25 and X.3," for complete details concerning configuration requirements for setting up X.25 encapsulation on protocol translators.

This capability is most useful on protocol translators with a serial network interface. From a user's point of view, connections are relatively transparent when making a remote connection over a PDN. In general, users simply make a connection request.

Example X.25 Connection Command:

```
ts>telnet far_host
```

If properly configured, the protocol translator takes this connection request, encapsulates the IP-based frame in an X.25 packet and sends it over the X.25 PDN to a specified location (based on the X.121 address). At the remote location, a Cisco router takes the request, de-encapsulates the X.25 packet, and puts the IP-based frame on the appropriate network. If all goes well, a virtual terminal connection is then made between the originating node and the target host.

X.3 PAD Connections

The protocol translator supports connections to an X.3 PAD host.

To make a connection to a PAD host, enter the **pad** command followed by the X.25 host address and press Return. The basic syntax is as follows:

pad *X.121-address*

If the X.25 host name has been defined with the **x25 host name** command, you can enter the host name rather than the host address. In addition, one-word PAD connections are supported, allowing you to simply enter the host name or X.121 address to make a connection.

When you are finished, logging off the host will end your connection. The EXEC terminates any connection after 10 minutes of inactivity or when you enter the **exit** command.

The **pad** command offers options that allow it to support X.3 functionality and to display the PAD parameters set by either a host or an application. These options and PAD Recommendation X.3 are discussed in the chapter, "Configuring X.25 and X.3."

Common User Level Commands

The following user level commands and facilities are applicable to all transmission protocols supported by Cisco protocol translators using the two-step translation method:

- The escape sequence capability
- **resume** command
- **login** command
- **lock** command
- **where** command
- **name-connection** command
- **disconnect** command
- **show sessions** command
- **show terminal** command
- **systat all** command
- **exit** and **quit** commands

Establishing Multiple Connections

You can have several concurrent connections open and switch back and forth between them. The number of connections that may be opened is defined by the **session-limit** command. Refer to the chapter, "Configuring the System." To switch between connections, first type the escape sequence, which by default is Ctrl-^, X (press the Ctrl, Shift and ^ key simultaneously, let go, then press the X key), to return to the system command prompt.

To make a new connection, use the procedure described in “Starting a Telnet Connection,” or “Starting an Rlogin Connection,” above. To get back to an existing connection, use the **resume** command, described next. Use the **where** command to list your open connections.

Switching Between Connections

The EXEC **resume** command returns to a previous connection. The general form is available with all connection protocols supported by the protocol translator. The general command syntax is as follows:

resume [*connection***]**

The optional argument *connection* is the connection name or number; the default is the most recent connection.

Note: Refer to the section, “Transmission Protocols,” for more information about **resume** command options available for specific protocols.

You can omit the command word **resume** and simply type the connection number to resume a connection. You can also return to the most recent session by just pressing the Return key.

Logging in as a Specific User

Use the EXEC command **login** to log in to a system with a specific user name. This command can be used to change your login name. One reason to change your login name is that an outgoing access list may depend on the specification of a particular user name. Refer to the discussion regarding Cisco’s TACACS (and other user authentication) facilities in the chapter, “Configuring the System.” In addition, IP access lists are discussed in the chapter, “Configuring TCP/IP.” The syntax for the **login** command is:

login

The system returns prompts for user name and password. If both are entered correctly, your session becomes associated with the specified user name. If there is no match, your connection to the terminal server reverts to the user name with which the **login** command attempt was made, if applicable. If no login name and password were originally required, the connection reverts to a session that is not associated with any name.

Example:

In the example below a user named *foouser* wants to change the login name being used to *sloan*. After entering the **login** command, the new name is entered along with an incorrect password. The system rejects the attempt to change user name. Next *foouser* attempts to change the login name to *klaus*. This time the correct password is entered and the user is now allowed access at the user level prompt under the user name of *klaus*.

```
sys-prompt>login

Username:sloan
Password:
% Access denied
Still logged in as "foouser"
sys-prompt>login

Username:klaus
Password:
sys-prompt>
```

Locking the Terminal

If you want to prevent access to your protocol translator session while keeping your connection sessions online, use the **lock** EXEC command. This command has the following syntax:

lock

The effect of this command is to lock your keyboard. It requires that the global configuration command **lockable** be included in your system configuration (described in the chapter, "Configuring the System"). The **lock** EXEC command remains in effect until the **clear line** privileged EXEC command (described in the chapter "Managing the System") is executed. If you are able to lock your session, you will be prompted to specify a password, which can be any arbitrary string. Enter the password you want to assign. The screen clears and displays the message "Locked." To regain access to your sessions, reenter the password. The protocol translator honors session timeouts on a locked line.

Listing Open Connections

The EXEC command **where** displays information about all open Telnet or rlogin connections associated with the current terminal line. The command has this simple syntax:

where

Following is a sample display:

```
cpt>where

Conn Host                Address            Byte  Idle Conn Name
   1 STUFT               131.108.19.50         0     0 STFUT
*  2 OTTER               192.31.7.24           0     0 OTTER
```

The information it displays includes the host name, address, number of characters waiting to be sent to the terminal, idle time, and connection name. An asterisk (*) indicates the current connection.

Naming a Connection

The **name-connection** command assigns a logical name to a connection. The EXEC prompts for the connection number and name to assign when you enter this command. The

where command displays a list of the assigned logical connection names.

The following example checks the connection number for the host *Eng1*, assigns the logical name *my host* to it, and then confirms the assignment.

```
cpt>where
Conn Host     Address         Byte  Idle  Conn  Name
*  1 Eng1     192.31.6.22        0     0  Eng1
   2 Term2    192.33.6.21        0     0  Term2
cpt>name-connection
Connection number:1
Enter logical name:  my host
Connection 1 to Eng1 will be named ”my host” [confirm]
cpt>where
Conn Host     Address         Byte  Idle  Conn  Name
*  1 Eng1     192.31.6.22        0     0  my host
```

Aborting a Connection

The EXEC command **disconnect** aborts a connection, and has this syntax:

disconnect [*connection***]**

The optional argument *connection* is a connection name or number; the default is the current connection. To list your open connection, use the **where** command.

The following example checks the open connections, and closes the connection to *Eng1*, which has the logical name of *my host*.

```
cpt>where
Conn Host     Address         Byte  Idle  Conn  Name
*  1 Eng1     192.31.6.22        0     0  my host
   2 Term2    192.33.6.21        0     0  Term2
cpt>disconnect my host
Closing connection to Eng1 [confirm]
```

Note: Do not use the **disconnect** command to end a session. Instead, log off the host, thus allowing the host to initiate the disconnect. If you cannot log off the host, use the **disconnect** command.

Displaying User Sessions

To display information about your active terminal sessions, use the **show sessions** EXEC command. This command has the following syntax:

show sessions

Example:

```
Conn Host                Address             Byte  Idle Conn Name
```

```
*  1 GUN                131.222.3.11          0     0   GUN
   2 BIG                131.222.3.14          0     5   BIG
```

The information it displayed includes the host name, address, number of characters waiting to be sent to the terminal, idle time, and connection name. An asterisk (*) indicates the users's current session.

Displaying Current Terminal Parameters

The EXEC command **show terminal** displays the configuration parameter settings for the current terminal line. Enter this command at the EXEC prompt:

show terminal

Following is a sample display of the command's output:

```
cpt>show terminal

Line 3, Location: "Library x1234", Type: ""
Length: 24 lines, Width: 80 columns
Baud rate (TX/RX) is 9600/9600, no parity, 2 stopbits, 8 databits
The escape character is "^^", followed by "x"
The local hold character is disabled
No flowcontrol in effect.
Status: Ready, Active
Capabilities: none
Idle EXEC timeout is 10 minutes.
Idle session timeout is not set.
Modem answer timeout is 15 seconds
Dispatch timeout is not set.
Allowed transports are telnet rlogin.  Preferred is telnet
Disconnect character is not set
Activation character is ^M (13)
No output characters are padded
Characters causing immediate data dispatching:
   Char    ASCII
```

The display includes a comprehensive report on the terminal settings in effect, including the preferred transport protocol.

Displaying Line Information

The EXEC command **systat** displays information about the active ports of the protocol translator, and has this syntax:

systat [all]

A sample display follows:

```
cpt>systat all

    Line    User      Host(s)           Idle  Location
   0 con 0                                    otter console
   1 tty 1                                    Bill  x1234
   2 tty 2            DAVE-SS2          1:32  Dave x1235
```

```
*  3 tty 3          STUFT              0   Denise x1236
   4 tty 4          incoming       11:17   remote.host.com
   5 tty 5                                 Sam- ROM emulator
   6 tty 6                                 CD4 ???
   7 tty 7                                 Teresa x1236
  10 tty 10         STUFT              3   Mark x1237
  11 tty 11         HEAT               1   Eng32-1
  12 tty 12         STUFT          17:32   CD8 ???
  13 tty 13                                1525: Sandy x1238
  14 tty 14         MEDDLE            23   Marsha x1239
  15 tty 15                                Randy x1240
  16 tty 16                                Kevin  x2120
  17 tty 17                                Robert x2141
  20 tty 20                                Bill  x2142
  21 tty 21                                Laser printer
  22 tty 22                                ""
```

The information it displays includes the line number, connection name, idle time, and terminal location. The optional keyword **all** requests information about both active and inactive ports.

Exiting a Session

The **exit** or **quit** command terminates the EXEC command parser and closes any active terminal sessions. Enter either command when you are finished with all sessions.