Chapter 1
Configuring TN3270

1

This chapter focuses on Cisco's support for IBM 3270 terminal emulation. It outlines the TN3270 terminal emulation environment, basic TN3270 terminal emulation operation with Cisco protocol translators, and configuration of Cisco protocol translators for TN3270 operation.

In brief, this section provides the following information:

- Cisco-specific TN3270 terminal emulation implementation issues
- Processes for making simple and customized terminal connections
- Configuration requirements for defining alternative terminal emulations
- Ttycap and keymap capabilities definitions and syntax descriptions
- A summary of global and line configuration commands relevant to Cisco's TN3270 terminal emulation

Note: TN3270 protocol translation support is disabled on an IGS system whenever routing or bridging is enabled.

Cisco's Implementation of TN3270

IBM's 3270 display terminals are among the computing community's most widely implemented and emulated environments for basic host-based computing. True IBM 3270-type terminals use a character format referred to as extended binary-coded decimal interchange code (EBCDIC). EBCDIC consists of 8-bit coded characters and was originally developed by IBM.

Cisco's TN3270 terminal emulation software is based on software developed by programmers at the University of California, Berkeley. This software allows any terminal to be used as an IBM 3270-type terminal. Users with non-3270 terminals can exploit the Cisco protocol translator's emulation capabilities to perform the functions of an IBM 3270-type terminal. Specifically, Cisco's implementation supports emulation of an IBM 3278-2 terminal. The protocol translator supports a 80 by 24 display.

Emulation is made possible by *termcap* and *curses* functions developed by Berkeley UNIX system developers. These functions translate the keyboard and terminal characteristics for ASCII-type terminals into those expected by an IBM host.

Note: ASCII characters are defined in the "ASCII Character Set" appendix.

Formally, a termcap is a two-part terminal handling mechanism. It consists of a database and a subroutine library. The database describes the capabilities of each terminal to be supported; the subroutine library allows programs to query the database and to make use of the capability values it contains.

For more information about defining termcaps refer to the following document:

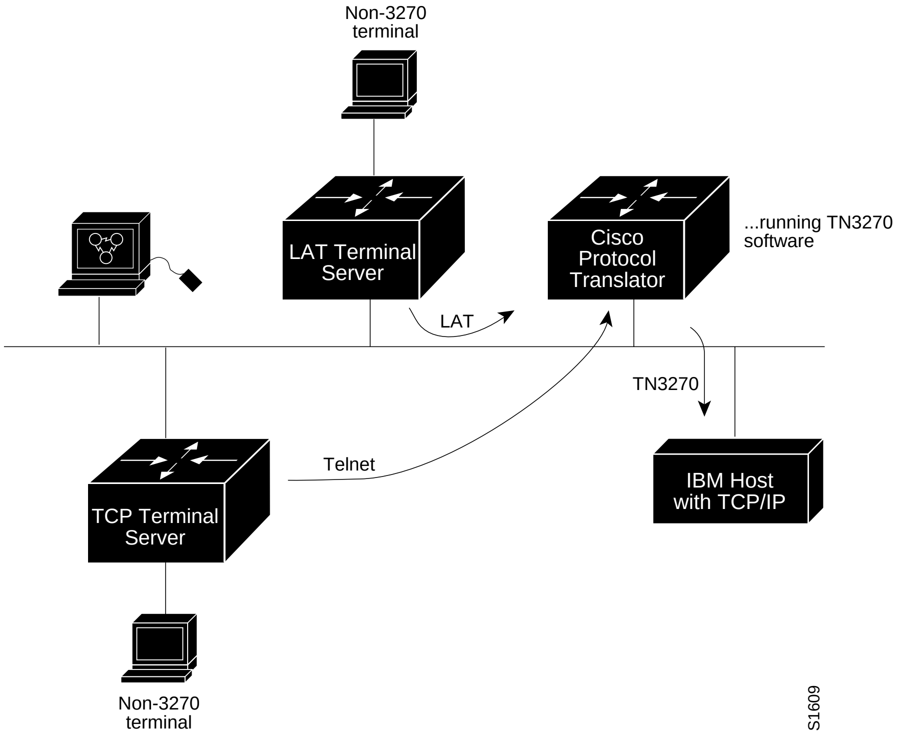
- *termcap & terminfo*, Jim Strang, Tim O'Reilly and Linda Mui. A Nutshell Handbook published by O'Reilly & Associates, Inc., 632 Petaluma Avenue, Sebastopol, CA, 95472. This technically-oriented guide provides concise details concerning each termcap characteristic and the definition of termcaps.

Cisco protocol translators include a default termcap entry (VT100). More samples are available directly from Cisco through an FTP file transfer from *ftp.cisco.com*.

Cisco's TN3270 emulation capability allows users to access an IBM host without using a special IBM server or a UNIX host acting as a server (see Figure 1-1). However, the IBM host must directly support TCP/IP, or have a front-end processor that supports TCP/IP.

Connection to IBM hosts from LAT, TCP, and X.25/PAD environments is accomplished via two-step translations. Refer to the "Using the Protocol Translator" chapter for more information about two-step translations. In general, Cisco's TN3270 support for protocol translators allows "outgoing" TN3270 connections only. In other words, LAT, TCP, and X.25/PAD users must first establish a connection with the protocol translator, then use the TN3270 facility from the protocol translator to make a connection to the IBM host.

Figure 1-1 Example 3270 Connection Environment



Making Simple Connections

If your network administrator has configured an appropriate default configuration for your serial line then the EXEC command **tn3270** is all you need to type 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:

```
tn3270 finance
```

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 “TN3270 Emulation Configuration” later in this chapter.

Connections Using Optional Configurations

For various reasons, the default terminal emulation may not be appropriate for your terminal. To override the default and obtain an optional *ttycap* and/or *keymap*, you must use the EXEC commands **terminal terminal-type** and/or **terminal keymap-type**.

Note: The ttycap and keymap must be included in the active Cisco protocol translator configuration image. To determine the presence of a ttycap and keymap, use the **show ttycap** and **show keymap** EXEC commands as described in "Examining Keymaps and TTYCAPs" provided later in this chapter.

Steps to adopt alternate configurations and start-up the protocol translator are outlined briefly below:

Step 1: Locally assign the terminal type.

Step 2: Locally assign the keyboard mapping.

Step 3: Establish the connection to the remote IBM host.

Using an Optional TTYCAP

Use the EXEC command **terminal terminal-type** to specify the current terminal type. This command has the following syntax:

terminal terminal-type *terminal-name*

The argument *terminal-name* is one of the terminal types specified within a protocol translator's configuration file. The elements of a terminal definition are discussed later in this chapter under "Configuring the Protocol Translator."

The following example results in the local protocol translator session adopting the VT100 ttycap entry.

Example:

```
terminal terminal-type VT100
```

Using Optional Keymaps

Use the EXEC command **terminal keymap-type** to locally specify the current keyboard type. This command has the following syntax:

terminal keymap-type *keymap-name*

The argument *keymap-name* is one of the keymap types specified within a protocol translator's configuration file. The elements of a keymap definition are discussed later in this chapter under "Configuring the Protocol Translator." The following example results in the local terminal adopting the keyboard mapping associated with the keymap entry named VT100map.

Example:

```
terminal keymap-type VT100map
```

Making the Connection

Once your local ttycap and keymap definitions have been specified, use the EXEC command **tn3270** as discussed in “Making Simple Connections” to start a 3270 session with a Cisco protocol translator. The syntax is as follows:

tn3270 *hostname*

TN3270 Emulation Configuration

To prepare a Cisco protocol translator for users requiring terminal and keyboard emulations that differ from the default definitions, you must complete some basic configuration steps.

Step 1: Determine what kind of terminal(s) you have.

Step 2: If necessary, set global configuration parameters **ttycap** and **keymap** (required if non-default ttycap is used).

Step 3: Set the line configuration parameters **terminal-type** and **keymap-type** to allow the protocol translator and target host to communicate correctly.

If you perform no special configuration of a new ttycap or keyboard, a Cisco protocol translator will automatically adopt a pre-defined VT100 emulation provided by Cisco. The associated keyboard is mapped as outlined in Table 1-2 later in this chapter. The default is illustrated in the example provided in “Keymap Syntax.” The default ttycap is illustrated in the example provided in “Loading a TTYCAP Over the Network.” If these configurations are acceptable, you can skip the configuration instructions addressed in “Defining New Keymaps and TTYCAPs” and “Assigning TTYCAP and Keymap Line Characteristics.”

Defining New Keymaps and TTYCAPs

You must perform global configuration procedures for TN3270 operation *only* when the default terminal and keyboard emulations are not acceptable for your terminal.

There are two alternatives to obtaining new keymaps and ttycaps:

- Obtain complete keymap and ttycap specifications as part of either a host-config or network-config files, and load them into the protocol translator configuration intact. To specify unique attributes, use a host-config file to obtain terminal characteristics. To specify a common set of attributes (characteristics applied to all terminal servers) use the network-config file to obtain terminal characteristics.
- Build keymaps and ttycaps on-line from the protocol translator command line interface.

When constructing keymaps and ttycaps, global configuration commands are available only through the **configure** command. Once you've accessed the system configuration facility, you can use the **keymap** and **ttycap** global configuration commands to define new keymaps and ttycaps.

Note: If you name a ttycap or keymap entry using the keyword *default*, the protocol translator will override the factory "default" and adopt the characteristics of the customer-defined "default."

Mapping the Keyboard

The following descriptions define the syntax for keymaps and address methods for including alternative keymap definitions within a Cisco protocol translator's configuration file.

Keymap Syntax

When emulating IBM-style 3270 terminals, a mapping must be performed between sequences of keys pressed at a user's (ASCII) keyboard, and the keys available on a 3270. For example, a 3270 has a key labeled EEOF which erases the contents of the current field from the location of the cursor to the end. In order to accomplish this function, the terminal user and a program emulating a 3270 must agree on what keys will be typed to invoke the function. The requirements for these sequences are:

- The first character of the sequence must be outside of the standard ASCII printable characters.
- No sequence may be a complete subset of another sequence (although sequences may share partial elements). Examples of acceptable and unacceptable entries are provided below.

Examples of Acceptable Keymap Entries:

```
pfk1 = '\E1';
pfk2 = '\E2';
```

Examples of Conflicting/Unacceptable Keymap Entries:

```
pfk1 = '\E1';
pfk11 = '\E11';
```

Note: The point is that in the acceptable example, the keymap entry for `pfk1` is not completely included in the keymap entry for `pfk2`. In contrast, with the unacceptable or conflicting keymap pair, the sequence used to represent `pfk1` is a complete subset of the sequence used to represent `pfk11`. Refer to the Example keymap entry provided later in this section for an example of how various keys can be represented to avoid this kind of conflict.

A keymap consists of an entry for a keyboard. The first part of an entry lists the names of the keyboards which use that entry. These names will often be the same as in ttycaps; however, often the terminals from various ttycap entries will all use the same keymap entry; for example, both 925 and 925vb (for 925 with visual bells) would probably use the same keymap entry. Additionally, there are occasions when it is necessary to specify a keyboard name as the name of the entry (for example, if a user requires a custom key layout).

After the names, separated by vertical bars (|), comes a left brace ({); the text that forms the definitions; and, finally, a right brace (}). Each definition consists of a reserved keyword (see Table 1-2 later in this chapter) which identifies the 3270 function, followed by an equal sign (=), followed by the various ways to generate this particular function, followed by a semi-colon (;). Each alternative way to generate the function is a sequence of ASCII characters enclosed inside single quotes (`''`); alternatives are separated by vertical bars (|). Inside the single quotes, a few characters are special. A caret (^) specifies that the next character is a control (Ctrl) character. So, the two-character string caret symbol-a (^a) represents Ctrl-a. The caret symbol-A sequence (^A) generates the same code as caret symbol-a (^a). To generate Delete (or DEL), enter the caret symbol-question mark (^?) sequence.

Note: The Ctrl-caret symbol combination (Ctrl-^), used to generate a hexadecimal 1E, is represented as two caret symbols in sequence (^^)—not as a caret-backslash-caret combination (^\^).

In addition to the caret, a letter may be preceded by a backslash (\ ). As this has little effect for most characters, its use is usually not recommended. For the case of a single quote (’), the backslash prevents that single quote from terminating the string. For the case of a caret (^), the backslash prevents the caret from having its special meaning. To have the backslash be part of the string, it is necessary to place two backslashes (\\) in the keymap. In addition, Table 1-1 lists special characters.

Table 1-1 Special Characters Supported by TN3270 Keymap Capability

Character	**Description**
`\E`	Escape character
`\n`	Newline
`\t`	Tab
`\r`	Carriage return

It is not necessary for each character in a string to be enclosed within single quotes. \E\E\E means three escape characters.

The following default entry is used by Cisco's TN3270 emulation software when it is unable to locate a valid keymap in the active configuration image. Table 1-2 lists the key names supported by the default Cisco TN3270 keymap.

Example Keymap Entry (Default):

```
ciscodefault{
    clear = '^z';
    flinp = '^x';
    enter = '^m';
    delete = '^d' | '^?';
    synch = '^r';
    reshow = '^v';
    eeof = '^e';
    tab = '^i';
    btab = '^b';
    nl = '^n';
    left = '^h';
    right = '^l';
    up = '^k';
    down = '^j';
    einp = '^w';
    reset = '^t';
    ferase = '^u';
    insrt = '\E ';
    pa1 = '^p1'; pa2 = '^p2'; pa3 = '^p3';
    pfk1 = '\E1'; pfk2 = '\E2'; pfk3 = '\E3'; pfk4 = '\E4';
    pfk5 = '\E5'; pfk6 = '\E6'; pfk7 = '\E7'; pfk8 = '\E8';
    pfk9 = '\E9'; pfk10 = '\E0'; pfk11 = '\E-'; pfk12 = '\E=';
    pfk13 = '\E!'; pfk14 = '\E@'; pfk15 = '\E#'; pfk16 = '\E$';
    pfk17 = '\E%'; pfk18 = '\E'; pfk19 = '\E&'; pfk20 = '\E*';
    pfk21 = '\E('; pfk22 = '\E)'; pfk23 = '\E_'; pfk24 = '\E+';
```

The following is the list of 3270 key names that are supported in this keymap. Note that some of the keys don't really exist on a 3270. In the following list, the starred (*) functions are not supported by TN3270. An unsupported function will cause the protocol translator to send a (possibly visual) bell sequence to the user's terminal.

To enter a keymap, provide a unique name for the keymap and explicitly define all special keys you intend to include in the keymap within curly brackets({}). Also, each line must be terminated with a backslash symbol (\), with the exception of the last line, which includes only the closing curly bracket symbol and an end-of-line character.

Table 1-2 3270 Key Names Supported by Default Keymap

3270 Key Name	**Functional Description**
*LPRT	Local print
DP	Dup character
FM	Field mark character
CURSEL	Cursor select
CENTSIGN	EBCDIC cent sign
RESHOW	Redisplay the screen
EINP	Erase input
EEOF	Erase end of field
DELETE	Delete character
INSRT	Toggle insert mode
TAB	Field tab
BTAB	Field back tab
COLTAB	Column tab
COLBAK	Column back tab
INDENT	Indent one tab stop
UNDENT	Undent one tab stop
NL	New line
HOME	Home the cursor
UP	Up cursor
DOWN	Down cursor
RIGHT	Right cursor
LEFT	Left cursor
SETTAB	Set a column tab
DELTAB	Delete a columntab
SETMRG	Set left margin
SETHOM	Set home position
CLRTAB	Clear all column tabs
*APLON	Apl on
*APLOFF	Apl off
*APLEND	Treat input as ASCII
*PCON	Xon/xoff on
*PCOFF	Xon/xoff off
DISC	Disconnect (suspend)
*INIT	New terminal type
*ALTK	Alternate keyboard dvorak
FLINP	Flush input
ERASE	Erase last character
WERASE	Erase last word
FERASE	Erase field
SYNCH	We are in synch with the user
RESET	Reset key-unlock keyboard
MASTER_RESET	Reset, unlock and redisplay
*XOFF	Please hold output
*XON	Please give me output
WORDTAB	Tab to beginning of next word
WORDBACKTAB	Tab to beginning of current/last word
WORDEND	Tab to end of current/next word
FIELDEND	Tab to last non-blank of current/next unprotected (writable) field.
PA1	Program attention 1
PA2	Program attention 2
PA3	Program attention 3

CLEAR	Local clear of the 3270 screen
TREQ	Test request
ENTER	Enter key
PFK1 to PFK30	Program function key 1 program function key 30

*Not supported by Cisco's TN3270 implementation

Table 1-3 illustrates the proper keys used to emulate each 3270 function when using the default key mapping supplied.

Table 1-3 Keys Used to Emulate Each 3270 Function with Default Keymap

Command Keys	**IBM 3270 Key**	**Default Key(s)**
	Enter	Return
	Clear	Ctrl-z
Cursor Movement Keys		
	New Line	Ctrl-n or Home
	Tab	Ctrl-i
	Back Tab	Ctrl-b
	Back Tab	Ctrl-b
	Cursor Left	Ctrl-h
	Cursor Right	Ctrl-l
	Cursor Up	Ctrl-k
	Cursor Down	Ctrl-j or LINE FEED
Edit Control Keys		
	Delete Char	Ctrl-d or RUB
	Erase EOF	Ctrl-e
	Erase Input	Ctrl-w
	Insert Mode	*ESC Space
	End Insert	ESC Space
Program Function Keys		
	PF1	ESC 1
	PF2	ESC 2
	...	...
	PF10	ESC 0
	PF11	ESC -
	PF12	ESC =
	PF13	ESC !
	PF14	ESC @
	...	...
	PF24	ESC +
Program Attention Keys		
	PA1	Ctrl-p 1
	PA2	Ctrl-p 2
	PA3	Ctrl-p 3
Local Control Keys		
	Reset After Error	Ctrl-r
	Purge Input Buffer	Ctrl-x
	Keyboard Unlock	Ctrl-t
	Redisplay Screen	Ctrl-v
Other Keys		
	Erase current field	Ctrl-u

*ESC refers to the Escape key.

Loading a Keymap over the Network

An alternate keymap can be loaded over the network as part of either host-config or network-config files. Refer to the chapter "Configuring the System" for more information about setting configuration file specifications and obtaining files over the network.

Note: In general, Cisco recommends building keymaps on a host and loading them over the network, as line by line construction using the **configure** mode lacks the flexibility of host-based editors.

Building Keymaps from Scratch

Use the global configuration command **keymap** to define specific characteristics of keyboard mappings. This command has the following syntax.

keymap *keymap-name keymap-entry*
no keymap *keymap-name*

The **keymap** command maps individual keys on a non-3270 keyboard to perform the function defined for the 3270 keyboard (refer to Table 1-2).

The argument *keymap-name* is a unique name that is used to identify a specific keymap.

The argument *keymap-entry* is the complete keymap configuration text, as illustrated in the previous discussion "Keymap Syntax."

Note: To enter a keymap you must enter a name (*ciscodefault* in the example provided in the preceding section) and explicitly define all special keys you intend to include in the keymap. Also, note that each line must be terminated with a backslash symbol (\), with the exception of the last line, which includes only a close brackets (}) symbol and an end-of-line character.

The command **no keymap** *keymap-name* removes the named keymap (if present) from the current image of the configuration file.

Example:

This example creates a custom keymap named *mainecap* that redefines the Enter key as a Return key.

```
keymap mainecap \
vt100{ \
enter = ‘^m’;\
}
```

Creating TTYCAPs

The following descriptions define the syntax for ttycaps and address methods for including alternative ttycap definitions within a Cisco protocol translator's configuration file.

TTYCAP Syntax

The following discussion defines the syntax used to create a ttycap entry in a Cisco protocol translator configuration file. The definition of the **ttycap** command entry in a configuration file is as follows:

ttycap *ttycap-name termcap-entry*
no ttycap *ttycap-name*

The argument *ttycap-name* can be up to 32 characters long. When used with the command **no ttycap**, the specified *ttycap-name* deletes any named ttycap entry from the configuration file.

The argument *termcap-entry* is the text that defines the termcap. The *termcap-entry* consists of two parts: a *name portion* and a *capabilities portion*. The *name portion* of a *termcap-entry* is a series of names that can be used to refer to a specific ttycap. Generally, these names should represent commonly recognized terminal names (such as VT100, VT200, etc.). Multiple names can be used. Each name is separated by a vertical bar symbol (|). The series is terminated by a colon symbol (:). The following example illustrates a name specification for a VT100 termcap.

Example Name Specification:

```
d0|vt100|vt100-am|vt100am|dec vt100:
```

The *capabilities portion* of the *termcap-entry* consists of a sequence of "termcap capabilities." These capabilities can be include boolean flags, string sequences, or numeric sequences; each individual capability is terminated using a colon symbol (:).

A *boolean flag* can be set to true by including the two-character capability name in the termcap entry. The absence of any supported flag results in the flag being set to false. The following is an example of a backspace boolean flag:

Example Boolean Flag:

```
bs:
```

A *string sequence* is a two-character capability name followed by an equal sign (=) and the character sequence. The following example illustrates the capability for homing the cursor:

Example String Sequence:

```
ho=\E[H:
```

The sequence \E represents the ESC character.

Note: Control characters can be represented in *string sequences* by entering a two-character sequence starting with a caret symbol (^), followed by the character to be used as a control character. The following example illustrates the definition of a control character.

Example Control Character String:

```
bc=^h:
```

In this example, the backspace is entered into the *termcap-entry* as the string sequence "^h."

A *numeric sequence* is a two-character capability name followed by an number symbol (#) and the number. The following example represents the number of columns on a screen.

Example Numeric Sequence:

```
co#80:
```

A definition of a ttycap for a VT100 terminal can be represented as illustrated in the following (partial) example.

Example TTYCAP Definition:

```
ttycap vt100 \
d0|vt100|vt100-am|vt100am|dec vt100:\
bs:ho=\E[H:co#80:
!
! This example is partial in nature and does
! not include the complete capabilities definition.
! It is provided for illustration only.
```

Note: The backslash symbol (\ ) can be used to extend the definition to multiple lines. The end of the ttycap *termcap-entry* is specified by a colon terminating a line followed by a end-of-line character and no backslash.

Loading a TTYCAP over the Network

An alternate ttycap can be loaded over the network as part of either host-config or network-config files. Refer to the "System Configuration" portion of this manual for more information concerning setting configuration file specifications and obtaining boot files over the network. The following example illustrates a typical keymap segment within the configuration file.

Note: In general, Cisco recommends building ttycaps on a host and loading them over the network, as line by line construction using the **configure** mode lacks the flexibility of host-based editors.

Example TTYCAP:

```
ttycap ttycap1\
  d0|vt100|vt100-am|vt100am|dec vt100:do=^J:co#80:li#24:\
  cl=50^[[;H^[[2J:bs:am:cm=5^[[%i%d;%dH:nd=2^[[C:up=2^[[A:\
  ce=3^[[K:so=2^[[7m:se=2^[[m:us=2^[[4m:ue=2^[[m:md=2^[[1m:\
  me=2^[[m:ho=^[[H:xn:sc=^[7:rc=^[8:cs=^[[%i%d;%dr:
```

Note: The administratively-definable ttycap capabilities illustrated in the preceding example and defined in Table 1-4 are case-sensitive.

Building TTYCAPs from Scratch

Use the global configuration command **ttycap** to define new ttycaps on your protocol translator on a line-by-line basis. This command has the following syntax.

ttycap *ttycap-name termcap-entry*
no ttycap *ttycap-name*

The argument *ttycap-name* is the unique name used to identify a specific ttycap.

The argument *termcap-entry* is the string of specific ttycap entries that define the characteristics of the ttycap, as illustrated in the previous discussion “Loading TTYCAPs over the Network.” The *termcap-entry* portion of that example is as follows:

Example Termcap-Entry:

```
d0|vt100|vt100-am|vt100am|dec vt100:do=^J:co#80:li#24:\
cl=50^[[;H^[[2J:bs:am:cm=5^[[%i%d;%dH:nd=2^[[C:up=2^[[A:\
ce=3^[[K:so=2^[[7m:se=2^[[m:us=2^[[4m:ue=2^[[m:md=2^[[1m:\
me=2^[[m:ho=^[[H:xn:sc=^[7:rc=^[8:cs=^[[%i%d;%dr:
```

The command **no ttycap** *name* removes the named ttycap (if present) from the current image of the configuration file. Refer to “TTYCAP Syntax” earlier in this section for more information about the syntax of ttycap entries.

Note: To enter a ttycap, you must enter a unique name or set of names (*ttycapl* in the example provided in the preceding section) and explicitly define all ttycap functions you intend to include in the ttycap. Also, note that each line must be terminated with a backslash symbol (\ ), with the exception of the last line, which includes only an end-of-line character.

Table 1-4 Definitions of TTYCAP Capabilities Supported by Cisco

Boolean Flags

am	Automatic margin
bs	Terminal can backspace with bs
ms	Safe to move in standout modes
nc	No currently working carriage return
xn	NEWLINE ignored after 80 cols (Concept)
xs	Standout not erased by overwriting (Hewlett-Packard)

String Sequences

AL	Add line below with cursor sequence
bc	Backspace if not ^h
bt	Backtab sequence
ce	Clear to end of line
cl	Clear screen, cursor to upper left
cm	Move cursor to row # and col #
cr	Carriage return sequence
cs	Change scrolling region
DL	Delete the line the cursor is on
ei	End insert mode
ho	Home, move cursor to upper left
ic	Character insert
im	Begin insert mode
is	initialization string (typically tab stop initialization)
ll	Move cursor to lower left corner
md	Turn on bold (extra bright) character attribute
me	Turn off all character attributes
nd	Non-destructive space
nl	Newline sequence
pc	Pad character if not NULL
rc	Restore cursor position
rs	resets terminal to known starting state
sc	Save cursor position
se	End standout mode (highlight)
so	Start standout mode (highlight)
ta	Tab
te	End programs that use cursor motion
ti	Initialization for programs that use cursor motion
uc	Underline character at cursor
ue	End underline mode
up	Move cursor up
us	Begin underline mode
vb	Visual bell
vs	Visual cursor
ve	Normal cursor

Number Sequences

li	Lines on the screen
co	Columns on the screen
sg	Standout glitch, number of spaces printed when entering or leaving standout display mode
ug	Underline glitch, number of spaces printed when entering or leaving underline mode

Assigning TTYCAP and Keymap Line Characteristics

If you intend to use an alternate ttycap and keymap, you *must* assign two characteristics associated with line configuration subcommands:

- terminal type
- keymap type

This information is used by the protocol translator when negotiating connections with hosts.

Specifying a Terminal Type for a Line

Use the line configuration subcommand **terminal-type** to specify the type of terminal connected to the line. This command must follow the corresponding **ttycap** global configuration entry in the configuration file. The **terminal-type** command has the following syntax:

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

The argument *terminal-name* is the name of a termcap defined within the configuration file. The TN3270 terminal-type negotiations use the specified terminal type when setting up a connection with the remote host. The argument *terminal-name* is the name associated with a specific ttycap. The command **no terminal-type** resets the terminal type for the line to the default (VT100).

Example:

The following example command sets the terminal type to VT220.

```
terminal-type VT220
```

Setting the terminal type to VT220 as shown requires that the ttycap be defined within the protocol translator's configuration either by obtaining a configuration file over the network that includes the ttycap definition or by defining the ttycap mapping via the global configuration **ttycap**.

Selecting a Keymap for a Line

Use the line configuration subcommand **keymap-type** to specify the keyboard mapping for a terminal connected to the line. This command must follow the corresponding **keymap** global configuration entry in the configuration file. The **keymap-type** command has the following syntax:

keymap-type *keymap name*
no keymap-type

The argument *keymap-name* is the name of a keymap defined within the configuration file of the protocol translator. The TN3270 terminal-type negotiations use the specified keymap type when setting up a connection with the remote host. The command **no keymap-type** resets the keyboard type for the line to the default (VT100).

The following example command sets the keyboard mapping to a keymap named `vt100map`.

Example:

```
   keymap-type vt100map
```

Setting the keyboard to a different keymap requires that a keymap be defined with the protocol translator's configuration either by obtaining a configuration file over the network that includes the keymap definition or by defining the keyboard mapping using the global configuration command **keymap**.

Notes on Start-up Sequence Priorities

When the protocol translator receives a **tn3270** EXEC command, it processes the request in the following sequences (for ttycaps and keymaps, respectively).

Choosing a TTYCAP

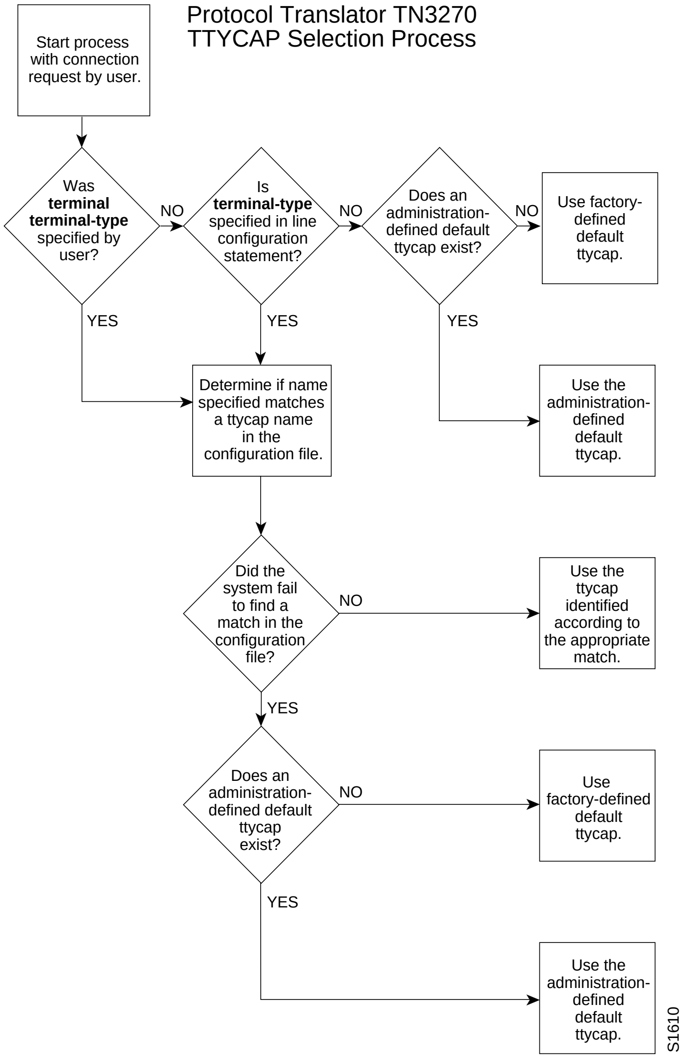
When attempting to select a ttycap search name, the protocol translator tries to adopt a valid ttycap by sequencing through the following set of prioritized decisions (illustrated in Figure 1-2):

1. Use the terminal name supplied by the user in the EXEC command **terminal terminal-type**.
2. Use the terminal name specified using the line configuration subcommand **terminal-type**.
3. Use the default ttycap supplied by the administrator.
4. Use the default vt100 ttycap provided by Cisco.

When the first or second priority checks fail (that is, the name specified does not match any name in the configuration file), the following rules apply:

Figure 1-2 illustrates the decision process used by the protocol translator to choose a ttycap for a specific TN3270 session.

Figure 1-2 Decision Diagram for Protocol Translator TTYCAP Selection Process


Protocol Translator TN3270
TTYCAP Selection Process
Start process with connection request by user.
Was terminal terminal-type specified by user?
NO
Is terminal-type specified in line configuration statement?
NO
Does an administration-defined default ttycap exist?
NO
Use factory-defined default ttycap.
YES
YES
YES
Determine if name specified matches a ttycap name in the configuration file.
Use the administration-defined default ttycap.
Did the system fail to find a match in the configuration file?
NO
Use the ttycap identified according to the appropriate match.
YES
Does an administration-defined default ttycap exist?
NO
Use factory-defined default ttycap.
YES
Use the administration-defined default ttycap.
S1610

- If a name is specified by the user, but fails to match any ttycap name in the configuration file, the protocol translator will adopt the “default” specified by the administrator. If one has not been specifically defined, the factory default ttycap is adopted.
- If a ttycap name is specified for line configuration (using the **terminal-type** line configuration command), that does not match any ttycap name in the configuration file, the protocol translator adopts the “default” specified by the administrator. If one has not been specifically defined, the factory defaults ttycap is adopted.

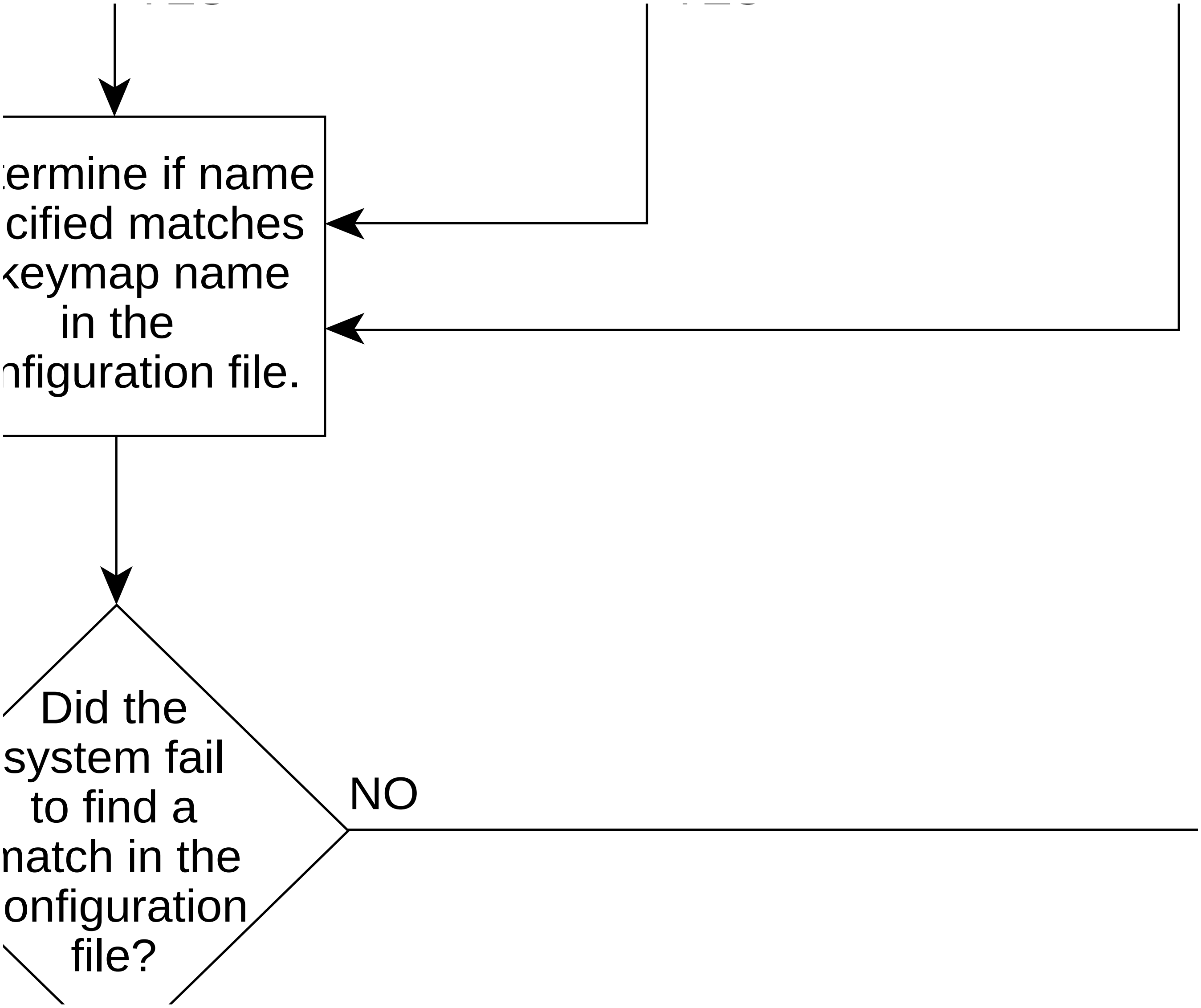
Choosing a Keymap

When attempting to select a keymap search name, the protocol translator tries to adopt a valid keymap by sequencing through the following set of prioritized decisions (illustrated in Figure 1-3):

1. Use the keymap name supplied by the user with the EXEC command **terminal keymap-type**.
2. Use the keymap name specified using the line configuration subcommand **keymap-type**.
3. Use the terminal name supplied by the user in the EXEC command **terminal terminal-type** to find a matching keymap.
4. Use the terminal name specified via the line configuration subcommand **terminal-type** to find a matching keymap.
5. Use the default keymap supplied by the administrator.
6. Use the default keymap supplied by Cisco.

Figure 1-3 illustrates the decision process used by the protocol translator to choose a keymap for a specific TN3270 session.

Figure 1-3 Decision Diagram for Protocol Translator Keymap Selection Process



When one of the first four priority checks fail (that is, the name specified does not match any name in the configuration file), the following rules apply:

- If a keymap name is specified by the user, but fails to match any keymap name in the configuration file, the protocol translator will adopt the “default” keymap specified by the administrator. If one has not been specifically defined, the factory default keymap is adopted.
- If a keymap name is specified for line configuration, but fails to match any keymap name in the configuration file, the protocol translator will adopt the “default” keymap specified by the administrator. If one has not been specifically defined, the factory default keymap is adopted.
- If a name is specified by the user using the EXEC command **terminal terminal-type**, but that name fails to match any keymap name in the configuration file, the protocol translator will adopt the “default” keymap specified by the administrator. If one has not been specifically defined, the factory default keymap is adopted.
- If a name is specified with the **terminal-type** line configuration command, but that

name fails to match any keymap name in the configuration file, the protocol translator will adopt the “default” keymap specified by the administrator. If one has not been specifically defined, the factory default keymap is adopted.

Examining Keymaps and TTYCAPs

If you’re not sure what ttycaps and keymaps are included in your system configuration, you can use the EXEC command **show** to list the ttycaps and keymaps available.

Listing Keymaps

The availability of a particular keymap is not tested until the connection takes place. Use the EXEC command **show keymap** to test for the availability of a keymap.

show keymap [*keymap-name***]**

If the optional [*keymap-name*] argument is left out, the keymap used for this terminal is shown.

If the optional [*keymap-name*] argument is included, the protocol translator searches for the specified keymap in its active configuration image, and lists the complete entry if found. If it is not found, an appropriate “not found” message is provided.

If the optional [*keymap-name*] argument is *all*, the system lists the names of all defined keymaps. The name of the default keymap is not listed.

The following example illustrates the screen display that results from the **show keymap** command.

Example:

```
ts_prompt>show keymap
ciscodefault { clear = '^z'; flinp = '^x'; enter = '^m';\
      delete = '^d' | '^?';\
      synch = '^r'; reshow = '^v'; eeof = '^e'; tab = '^i';\
      btab = '^b'; nl = '^n'; left = '^h'; right = '^l';\
      up = '^k'; down = '^j'; einp = '^w'; reset = '^t';\
      xoff = '^s'; xon = '^q'; escape = '^c'; ferase = '^u';\
      insrt = '\E ';\
      pa1 = '^p1'; pa2 = '^p2'; pa3 = '^p3';\
      pfk1 = '\E1'; pfk2 = '\E2'; pfk3 = '\E3'; pfk4 = '\E4';\
      pfk5 = '\E5'; pfk6 = '\E6'; pfk7 = '\E7'; pfk8 = '\E8';\
      pfk9 = '\E9'; pfk10 = '\E0'; pfk11 = '\E-'; pfk12 = '\E=';\
      pfk13 = '\E!'; pfk14 = '\E@'; pfk15 = '\E#'; pfk16 = '\E$';\
      pfk17 = '\E%'; pfk18 = '\E\^'; pfk19 = '\E&'; pfk20 = '\E*';\
      pfk21 = '\E('; pfk22 = '\E)'; pfk23 = '\E_'; pfk24 = '\E+';\
}
```

Listing TTYCAPs

The availability of a particular ttycap is not tested until the connection takes place. To test for the availability of a ttycap, use the EXEC command **show ttycap**. This command has the following syntax:

show ttycap [*ttycap-name***]**

If the optional [*ttycap-name*] argument is left out, the ttycap used for this terminal is shown.

If the optional [*ttycap-name*] argument is included, the protocol translator searches for the specified ttycap in its active configuration image, and lists the complete entry if found. If it is not found, an appropriate “not found” message is provided.

If the optional [*ttycap-name*] argument is *all*, the system lists the names of all defined ttycaps. The name of the default ttycap is not listed.

The following examples illustrate several screen displays that result from the various **show ttycap** command options.

Examples:

```
ts_prompt>show ttycap
d0|vt100|vt100-am|vt100am|dec vt100:do=^J:co#80:li#24:\
cl=50^[[;H^[[2J:bs:am:cm=5^[[%i%d;%dH:nd=2^[[C:up=2^[[A:\
ce=3^[[K:so=2^[[7m:se=2^[[m:us=2^[[4m:ue=2^[[m:md=2^[[1m:\
me=2^[[m:ho=^[[H:xn:sc=^[7:rc=^[8:cs=^[[%i%d;%dr:

ts_prompt>show ttycap all
ttycap3    d0|vt100|vt100-am|vt100am|dec vt100
ttycap2    dl|vt200|vt220|vt200-js|vt220-js|dec vt200 series with jump scroll
ttycap1    ku|h19-u|h19u|heathkit with underscore cursor

ts_prompt>show ttycap ttycap1
ttycap1  ku|h19-u|h19u|heathkit with underscore cursor:\:vs@:ve@:tc=h19-b:\
        :al=1*\EL:am:le=^H:bs:cd=\EJ:ce=\EK:cl=\EE:cm=\EY%+ %+\
        :co#80:dc=\EN:\:dl=1*\EM:do=\EB:ei=\EO:ho=\EH\
        :im=\E@:li#24:mi:nd=\EC:as=\EF:ae=\EG:\
        :ms:pt:sr=\EI:se=\Eq:so=\Ep:up=\EA:vs=\Ex4:ve=\Ey4:\
        :kb=^h:ku=\EA:kd=\EB:kl=\ED:kr=\EC:kh=\EH:kn#8:ke=\E>:ks=\E=:\
        :k1=\ES:k2=\ET:k3=\EU:k4=\EV:k5=\EW:\
        :l6=blue:l7=red:l8=white:k6=\EP:k7=\EQ:k8=\ER:\
        :es:hs:ts=\Ej\Ex5\Ex1\EY8%+ \Eo:fs=\Ek\Ey5:ds=\Ey1:
```

TN3270 Global Command Summary

The following TN3270 global configuration commands specify system-wide parameters for 3270 terminal emulation support.

[no] keymap *keymap-name keymap-entry*

Defines specific characteristics of keyboard mappings. The **keymap** command maps individual keys on a non-3270 keyboard to perform the function defined for the 3270 keyboard.

[no] ttycap *ttycap-name termcap-entry*

Defines new ttycaps on your protocol translator on a per-terminal basis. The argument *text* is the string of specific ttycap entries that define the characteristics of the ttycap.

TN3270 Line Configuration Subcommand Summary

The following line configuration subcommands define line-specific parameters for TN3270 support.

[no] keymap-type *keymap-name*

Specifies the keyboard mapping for a terminal connected to the line. The argument *keymap-name* is the pre-defined name of a keymap defined in the protocol translator configuration. The TN3270 terminal-type negotiations use the specified keymap type when setting up a connection with the remote host. The command **no keymap-type** resets the keyboard type for the line to the default (VT100).

[no] terminal-type *ttycap-name*

Specifies the type of terminal connected to the line. The TN3270 terminal-type negotiations use the specified terminal type when setting up a connection with the remote host. The command **no terminal-type** resets the terminal type for the line to the default (VT100).