CHAPTER 13

TN3270 Configuration Commands

TN3270 terminal emulation software allows any terminal to be used as an IBM 3270-type terminal. Users with non-3270 terminals can take advantage of the emulation capabilities to perform the functions of an IBM 3270-type terminal. Specifically, Cisco's implementation supports emulation of an IBM 3278-2 terminal providing an 80-by-24 display.

Use the commands in this chapter to configure and monitor TN3270 connections. For configuration information and examples, refer to the *Access and Communication Servers Configuration Guide*. For information about making connections to IBM 3278 hosts, refer to the *Cisco Access Connection Guide*.

keymap

To define specific characteristics of keyboard mappings, use the **keymap** global configuration command. To remove the named keymap from the current image of the configuration file, use the **no** form of this command.

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

Syntax Description

keymap-name	Name of the file containing the keyboard mappings. The name can be up to 32 characters long and must be unique.
keymap-entry	Commands that define the keymap.

Default

VT100 keyboard emulation

Command Mode

Global configuration

Usage Guidelines

The **keymap** command maps individual keys on a non-3270 keyboard to perform the function defined for the 3270 keyboard. Use the EXEC command **show keymap** to test for the availability of a keymap.

The guidelines for creating a keymap file follow.

Do not name a ttycap entry filename *default* or the communication server will adopt the newly defined entry as the default.

The Keymap Entry Structure

A keymap is a keyboard map file. A keymap consists of an entry for a keyboard. The first part of keymap lists the names of the keyboards that use that entry. These names will often be the same as in the ttycaps (terminal emulation) file, and often the terminals from various ttycap entries will use the same keymap entry. For example, both 925 and 925vb (for 925 with visual bells) terminals would probably use the same keymap entry. There are other circumstances in which 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, which are separated by vertical bars (|), comes a left brace ({), the text that forms the definitions, and a right brace (}), as follows:

```
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+';\
}
```

Each definition consists of a reserved keyword, which identifies the 3270 function, followed by an equal sign (=), followed by the various ways to generate this particular function, followed by a semicolon (;), as follows:

```
pa1 = '^p1'; pa2 = '^p2'; pa3 = '^p3';\
```

Each alternative way to generate the function is a sequence of ASCII characters enclosed inside single quotes (‘’); the alternatives are separated by vertical bars (|), as follows:

```
delete = '^d' | '^?';\
```

Inside the single quotes, a few characters are special. A caret (^) specifies that the next character is a control (Ctrl) character. The two-character string caret-a (^a) represents Ctrl-a. The caret-A sequence (^A) generates the same code as caret-a (^a). To generate Delete (or DEL), enter the caret-question mark (^?) sequence.

Note The Ctrl-caret 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 can be preceded by a backslash (\). Because this has little effect for most characters, its use is usually not recommended. In the case of a single quote (’), the backslash prevents that single quote from terminating the string. In the case of a caret (^), the backslash prevents the caret from having its special meaning. To include the backslash in the string, place two backslashes (\\) in the keymap. Table 13-1 lists other supported special characters.

Table 13-1 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. For example, \E\E\E means three escape characters.

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

Keymap Restrictions

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-type keyboard. For example, a 3270-type keyboard has a key labeled EEOF that erases the contents of the current field from the location of the cursor to the end. To accomplish this function, the terminal user and a program emulating a 3270-type keyboard must agree on what keys will be typed to invoke the function. The requirements for these sequences follow:

- The first character of the sequence must be outside of the standard ASCII printable characters.
- No sequence can be a complete subset of another sequence (although sequences can share partial elements).

Following are examples of acceptable keymap entries:

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

Following are examples of unacceptable keymap entries:

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

In the acceptable example, the keymap entry for *pfk1* is not completely included in the keymap entry for *pfk2*. By contrast, in 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 keymap entry provided later in this section for an example of how various keys can be represented to avoid this kind of conflict.

Table 13-2 lists 3270 key names that are supported in this keymap. Note that some of the keys do not really exist on a 3270-type keyboard. An unsupported function will cause the communication server to send a (possibly visual) bell sequence to the user's terminal.

Table 13-2 3270 Key Names Supported by Default Keymap

3270 Key Name	Functional Description
LPRT[1]	Local print
DP	Duplicate character
FM	Field mark character
CURSEL	Cursor select
CENTSIGN	EBCDIC cent sign
RESHOW	Redisplay the screen
EINP	Erase input
EEOF	Erase end of field

3270 Key Name	Functional Description
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 column tab
SETMRG	Set left margin
SETHOM	Set home position
CLRTAB	Clear all column tabs
APLON[1]	Apl on
APLOFF[1]	Apl off
APLEND[1]	Treat input as ASCII
PCON[1]	Xon/xoff on
PCOFF[1]	Xon/xoff off
DISC	Disconnect (suspend)
INIT[1]	New terminal type
ALTK[1]	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[1]	Please hold output
XON[1]	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

3270 Key Name	Functional Description
FIELDEND	Tab to last nonblank 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

1. Not supported by Cisco's TN3270 implementation

Table 13-3 illustrates the proper keys used to emulate each 3270 function when using default key mappings.

Table 13-3 Keys Used to Emulate Each 3270 Function with Default Keymap

Key Types	IBM 3270 Key	Default Keys
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[1]
	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 +

Key Types	IBM 3270 Key	Default Keys
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	Enter	Return
	Clear	Ctrl-z
	Erase current field	Ctrl-u

1. ESC refers to the Escape key.

Example

The following example is the default entry used by the TN3270 emulation software when it is unable to locate a valid keymap in the active configuration image. Table 13-2 lists the key names supported by the default Cisco TN3270 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';\
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+';\
}
```

Related Commands

keymap-type
show keymap
terminal-type

keymap-type

To specify the keyboard map for a terminal connected to the line, use the **keymap-type** line configuration command. To reset the keyboard type for the line to the default, use the **no** form of this command.

keymap-type *keymap-name*
no keymap-type

Syntax Description

keymap-name	Name of a keymap defined within the configuration file of the communication server. The TN3270 terminal-type negotiations use the specified keymap type when setting up a connection with the remote host.

Default

VT100

Command Mode

Line configuration

Usage Guidelines

This command must follow the corresponding **keymap** global configuration entry in the configuration file. The TN3270 terminal-type negotiations use the specified keymap type when setting up a connection with the remote host.

Setting the keyboard to a different keymap requires that a keymap be defined with the communication server'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**.

Use the EXEC command **show keymap** to test for the availability of a keymap.

Example

The following example sets the keyboard mapping to a keymap named *vt100map*:

```
line 3
keymap-type vt100map
```

Related Commands

keymap
show keymap
ttycap

show keymap

Use the **show keymap** EXEC command to test for the availability of a keymap after a connection on a communication server takes place.

show keymap [*keymap-name* | **all**]

Syntax Description

keymap-name	(Optional) Name of the keymap.
all	(Optional) Lists the names of all defined keymaps. The name of the default keymap is not listed.

Command Mode

EXEC

Usage Guidelines

The communication server searches for the specified keymap in its active configuration image and lists the complete entry if found. If the keymap is not found, an appropriate “not found” message appears.

If you do not use any arguments with the **show keymap** command, then the keymap currently used for the terminal is displayed.

Sample Display

The following is sample output from the **show keymap** command:

```
cs# 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+';\
}
```

show tn3270 ascii-hexval

To determine ASCII-hexadecimal character mappings, use the **show tn3270 ascii-hexval** EXEC command.

show tn3270 ascii-hexval

Syntax Description

This command has no arguments or keywords.

Command Mode

EXEC

Usage Guidelines

Use the **show tn3270 ascii-hexval** command to display the hexadecimal value of a character on your keyboard. After entering the **show tn3270 ascii-hexval** command, you are prompted to press a key. The hexadecimal value of the ASCII character is displayed. This command is useful for users who do not know the ASCII codes associated with various keys or do not have manuals for their terminals.

Examples

The following examples show how the **show tn3270 ascii-hexval** command works:

```
cs> show tn3270 ascii-hexval
Press key> 7 - hexadecimal value is 0x37.

chaff> show tn3270 ascii-hexval
Press key> f - hexadecimal value is 0x66.

tarmac> show tn3270 ascii-hexval
Press key> not printable - hexadecimal value is 0xD.
```

Related Commands

tn3270 character-map
show tn3270 character-map

show tn3270 character-map

To display character mappings between ASCII and EBCDIC, use the **show tn3270 character-map** EXEC command.

show tn3270 character-map {**all** | *ebcdic-in-hex*}

Syntax Description

all	Displays all nonstandard character mappings.
ebcdic-in-hex	Displays the ASCII mapping for a specific EBCDIC character.

Command Mode

EXEC

Sample Display

The following is sample output from the **show tn3270 character-map** command:

```
cs# show tn3270 character-map all

EBCDIC 0x81 <=> 0x78 ASCII
EBCDIC 0x82 <=> 0x79 ASCII
EBCDIC 0x83 <=> 0x7A ASCII
```

Related Commands

tn3270 character-map
show tn3270 ascii-hexval

show ttycap

To test for the availability of a ttycap after a connection on a communication server takes place, use the **show ttycap** EXEC command.

show ttycap [*ttycap-name* | **all**]

Syntax Description

ttycap-name	(Optional) Name of a ttycap.
all	(Optional) Lists the names of all defined ttycaps. The name of the default ttycap is not listed.

Command Mode

EXEC

Usage Guidelines

The communication server 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 appear.

If you do not include any arguments with the **show ttycap** command, then the current keymap used for the terminal is displayed.

Sample Display

The following is sample output from the **show ttycap** command:

```
cs# 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:

cs# 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

cs# 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:
```

terminal-type

To specify the type of terminal connected to the line, use the **terminal-type** line configuration command. To reset the terminal type for the line to the default, use the **no** form of this command.

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

Syntax Description

terminal-name	Name of a termcap defined within the configuration file

Default

VT100

Command Mode

Line configuration

Usage Guidelines

The **terminal-type** command must follow the corresponding **ttycap** global configuration entry in the configuration file. Use the EXEC command **show ttycap** to test for the availability of a ttycap.

The TN3270 terminal-type negotiations use the specified terminal type when setting up a connection with the remote host.

Setting the terminal type to VT220 requires that the ttycap be defined within the communication server's configuration either by obtaining a configuration file over the network that includes the ttycap definition, or by defining the ttycap mapping via the **ttycap** global configuration command.

Example

The following example command sets the terminal line 5 to type *VT220*:

```
line 5
terminal-type VT220
```

Related Commands

keymap
show ttycap
ttycap

tn3270 8bit display

To configure the communication server to use the mask set by the **data-character-bits** {**7** | **8**} line configuration command or the **terminal data-character bits** {**7** | **8**} EXEC command, use the **tn3270 8bit display** line configuration command. To restore the default 7-bit mask used for TN3270 connections, use the **no** form of this command.

tn3270 8bit display
no tn3270 8bit display

Syntax Description

This command has no arguments or keywords.

Default

Disabled

Command Mode

Line configuration

Usage Guidelines

Use the **tn3270-character-map** command to map between extended EBCDIC or extended ASCII characters.

Example

The following example configures the communication server to use the mask set by the **data-character-bits** line configuration and EXEC commands on line 5:

```
line 5
tn3270 8bit display
```

Related Commands

A dagger (†) indicates that the command is documented in another chapter. Two daggers (††) indicate that the command is documented in the *Cisco Access Connection Guide*.

data-character-bits {**7** | **8**}†
terminal data-character-bits {**7** | **8**} ††

tn3270 8bit transparent-mode

To configure the communication server to use the mask set by the **data-character-bits** {**7** | **8**} line configuration command or the **terminal data-character bits** {**7** | **8**} EXEC command, use the **tn3270 8bit display** line configuration command. To restore the default 7-bit mask used for TN3270 connections, use the **no** form of this command.

tn3270 8bit transparent-mode
no tn3270 8bit transparent-mode

Syntax Description

This command has no arguments or keywords.

Default

Disabled

Command Mode

Line configuration

Usage Guidelines

This command is needed if you are using a file transfer protocol such as Kermit in 8-bit mode or you are using 8-bit graphics, both of which rely on transparent mode.

Example

The following example configures the communication server to use the mask set by the **data-character-bits** line configuration and EXEC commands on line 5:

```
line 5
tn3270 8bit transparent-mode
```

Related Commands

A dagger (†) indicates that the command is documented in another chapter. Two daggers (††) indicate that the command is documented in the *Cisco Access Connection Guide*.

data-character-bits {**7** | **8**}†
terminal data-character-bits {**7** | **8**}

tn3270 character-map

To create a two-way binding between EBCDIC and ASCII characters, use the **tn3270 character-map** global configuration command. To restore default character mappings, use the **no** form of this command.

tn3270 character-map *ebcdic-in-hex ascii-in-hex*
no tn3270 character-map {**all** | *ebcdic- in-hex*} [*ascii-in-hex*]

Syntax Description

ebcdic-in-hex	Hexadecimal value of an EBCDIC character.
ascii-in-hex	Hexadecimal value of an ASCII character.
all	Indicates all character mappings.

Default

Disabled

Command Mode

Global configuration

Usage Guidelines

Use this command to print international characters that are EBCDIC characters not normally printed, including umlauts (¨) and tildes (~). The command first restores default mapping for both EBCDIC and ASCII characters. In the **no** form of the command, the **all** keyword resets all character mappings to Cisco defaults.

Table 13-4 shows the default character mappings between ASCII and EBCDIC in decimal and hexadecimal format.

Table 13-4 Default ASCII, EBCDIC Character Mappings

Character	ASCII Decimal	ASCII Hexadecimal	EBCDIC Decimal	EBCDIC Hexadecimal
!	33	0x21	90	0x5a
"	34	0x22	127	0x7f
#	35	0x23	123	0x7b
$	36	0x24	91	0x5b
%	37	0x25	108	0x6c
&	38	0x26	80	0x50
’	39	0x27	125	0x7d
(	40	0x28	77	0x4d
)	41	0x29	93	0x5d
*	42	0x2a	92	0x5c

Character	ASCII Decimal	ASCII Hexadecimal	EBCDIC Decimal	EBCDIC Hexadecimal
+	43	0x2b	78	0x4e
,	44	0x2c	107	0x6b
-	45	0x2d	96	0x60
.	46	0x2e	75	0x4b
/	47	0x2f	97	0x61
0	48	0x30	240	0xf0
1	49	0x31	241	0xf1
2	50	0x32	242	0xf2
3	51	0x33	243	0xf3
4	52	0x34	244	0xf4
5	53	0x35	245	0xf5
6	54	0x36	246	0xf6
7	55	0x37	247	0xf7
8	56	0x38	248	0xf8
9	57	0x39	249	0xf9
:	58	0x3a	122	0x7a
;	59	0x3b	94	0x5e
<	60	0x3c	76	0x4c
=	61	0x3d	126	0x7e
>	62	0x3e	110	0x6e
?	63	0x3f	111	0x6f
@	64	0x40	124	0x7c
A	65	0x41	193	0xc1
B	66	0x42	194	0xc2
C	67	0x43	195	0xc3
D	68	0x44	196	0xc4
E	69	0x45	197	0xc5
F	70	0x46	198	0xc6
G	71	0x47	199	0xc7
H	72	0x48	200	0xc8
I	73	0x49	201	0xc9
J	74	0x4a	209	0xd1
K	75	0x4b	210	0xd2
L	76	0x4c	211	0xd3
M	77	0x4d	212	0xd4
N	78	0x4e	213	0xd5
O	79	0x4f	214	0xd6
P	80	0x50	215	0xd7

Character	ASCII Decimal	ASCII Hexadecimal	EBCDIC Decimal	EBCDIC Hexadecimal
Q	81	0x51	216	0xd8
R	82	0x52	217	0xd9
S	83	0x53	226	0xe2
T	84	0x54	227	0xe3
U	85	0x55	228	0xe4
V	86	0x56	229	0xe5
W	87	0x57	230	0xe6
X	88	0x58	231	0xe7
Y	89	0x59	232	0xe8
Z	90	0x5a	233	0xe9
[	91	0x5b	173	0xad
\	92	0x5c	224	0xe0
]	93	0x5d	189	0xbd
^	94	0x5e	95	0x5f
_	95	0x5f	109	0x6d
`	96	0x60	121	0x79
a	97	0x61	129	0x81
b	98	0x62	130	0x82
c	99	0x63	131	0x83
d	100	0x64	132	0x84
e	101	0x65	133	0x85
f	102	0x66	134	0x86
g	103	0x67	135	0x87
h	104	0x68	136	0x88
i	105	0x69	137	0x89
j	106	0x6a	145	0x91
k	107	0x6b	146	0x92
l	108	0x6c	147	0x93
m	109	0x6d	148	0x94
n	110	0x6e	149	0x95
o	111	0x6f	150	0x96
p	112	0x70	151	0x97
q	113	0x71	152	0x98
r	114	0x72	153	0x99
s	115	0x73	162	0xa2
t	116	0x74	163	0xa3
u	117	0x75	164	0xa4
v	118	0x76	165	0xa5

Character	ASCII Decimal	ASCII Hexadecimal	EBCDIC Decimal	EBCDIC Hexadecimal
w	119	0x77	166	0xa6
x	120	0x78	167	0xa7
y	121	0x79	168	0xa8
z	122	0x7a	169	0xa9
{	123	0x7b	192	0xc0
|	124	0x7c	79	0x4f
}	125	0x7d	208	0xd0
~	126	0x7e	161	0xa1

Example

The following example creates a two-way binding between an EBCDIC character and an ASCII character:

```
tn3270 character-map 0x81 0x78
```

Related Commands

show tn3270 character-map
show tn3270 ascii-hexval

tn3270 datastream

Use the **tn3270 datastream extended** global configuration command to enable the TN3270 extended datastream. Use the **no** form of the command to return to the normal TN3270 datastream.

tn3270 datastream [**extended** | **normal]**
no tn3270 datastream

Syntax Description

extended	Extended datastream.
normal	Normal datastream.

Default

Normal datastream

Command Mode

Global configuration

Usage Guidelines

This command causes an "-E" to be appended to the terminal type string sent to the IBM host. This allows you to use the extended TN3270 features.

Example

```
cserver(config)#tn3270 datastream ?
  extended  Use extended TN3270 datastream
  normal    Use normal TN3270 datastream
```

tn3270 null-processing

Use the **tn3270 null-processing** global configuration command to specify how NULLs are handled. Use the no form of the command to return to 7171 NULL processing.

tn3270 null-processing [**3270** | **7171**]
no tn3270 null-processing [**3270** | **7171**]

Syntax Description

3270	NULLs are compressed out of the string, as on a 3278-x terminal.
7171	NULLs are converted to spaces, as on a 7171 controller.

Default

7171 NULL processing

Command Mode

Global configuration

Usage Guidelines

If a user enters data, uses an arrow key to move the cursor to the right on the screen, and then enters more data, the intervening spaces are filled with NULLs. To specify how NULLs are handled, enter the command **tn3270 null-processing** either with the argument **3270**, where NULLs are compressed out of the string (as on a real 3278-x terminal) or the argument **7171**, where NULLs are converted to spaces as on a 7171 controller. Enter this command in global configuration.

Example

```
cserver(config)#tn3270 null-processing ?
  3270  Use 3270-style null processing
  7171  Use 7171-style null processing
```

tn3270 reset-required

Use the **tn3270 reset-required** global configuration command to lock a terminal after input error until the user resets the terminal. Use the no form of the command to return to the default of no reset required.

tn3270 reset-required
no tn3270 reset-required

Syntax Description

This command has no arguments or keywords.

Default

No reset is required

Command Mode

Global configuration

Usage Guidelines

On a 3278-x terminal, the keyboard is locked and further input is not permitted after input error (due to field overflow, invalid entry, and so on.), until the user presses the RESET key. Most TN3270 implementations leave the keyboard unlocked and remove any error message on the next key input after the error. Use this command to lock the keyboard until the user performs a reset.

ttycap

To define characteristics of a terminal emulation file, use the **ttycap** global configuration command. To delete any named ttycap entry from the configuration file, use the **no** form of this command.

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

Syntax Description

ttycap-name	Name of a file. It can be up to 32 characters long and must be unique.
termcap-entry	Commands that define the tttycap. Consists of two parts (see Usage Guidelines for details).

Default

VT100 terminal emulation

Command Mode

Global configuration

Usage Guidelines

Use the EXEC command **show ttycap** to test for the availability of a ttycap.

Note Do not type a ttycap entry filename “default” or the communication server will adopt the newly defined entry as the default.

The *termcap-entry* consists of two parts: a *name* portion and a *capabilities* portion:

- The *name* portion is a series of names that can be used to refer to a specific terminal type. Generally, these names should represent commonly recognized terminal names (such as VT100 and VT200). 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.

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

- The *capabilities* portion of the termcap-entry consists of a sequence of termcap capabilities. These capabilities can 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:

  ```
  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:

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

The sequence \E represents the ESC character.

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.

```
bc=^h:
```

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

```
co#80:
```

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

For the definitions of supported Boolean-flag ttycap capabilities, see Table 13-5. For the definitions of supported string-sequence ttycap capabilities, see Table 13-6. For the definitions of supported number-sequence ttycap capabilities, see Table 13-7.

Table 13-5 Definitions of Ttycap Capabilities: Boolean Flags

Boolean Flag	Description
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 columns (Concept)
xs	Standout not erased by overwriting (Hewlett-Packard)

Table 13-6 Definitions of Ttycap Capabilities: String Sequences

String Sequence	Description
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

String Sequence	Description
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	Nondestructive 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

Table 13-7 Definitions of Ttycap Capabilities: Number Sequences

Number Sequence	Description
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

Example

The following is an example of a ttycap file. Refer to the chapter "Configuring TN3270" in the *Access and Communication Servers Configuration Guide* publication and the *tn3270.examples* file in the Cisco *ftp@cisco.com* directory for more examples.

```
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:
```

Related Commands

terminal-type
keymap-type