John Day
June 27,1977

Telnet Data Entry Terminal Option

1. Command Name and code:

    DET 20

2. Command Meanings

IAC WILL DET
The sender of this command REQUESTS or AGREES to send and receive subcommands to control the Data Entry Terminal.

IAC WONT DET
The sender of this command REFUSES to send and receive subcommands to control the Data Entry Terminal.

IAC DO DET
The sender of this command REQUESTS or AGREES to send and receive subcommands to control the Data Entry Terminal.

IAC DONT DET
The sender of this command REFUSES to send and receive subcommands to control the Data Entry Terminal.

The DET option uses five classes of subcommands 1) to establish the requirements and capabilities of the application and the terminal, 2) to format the screen, and to control the 3) edit, 4) erasure, and 5) transmission functions. The subcommands that perform these functions are described below.

The Network Virtual Data Entry Terminal(NVDET)

The NVDET consists of a keyboard and a rectangular display. The keyboard is capable of generating all of the characters of the ASCII character set. In addition, the keyboard may possess a number of function keys which when pressed cause a FN subcommand to be sent. (Although most DET's will support one or more peripheral devices such as a paper tape reader or a printer, this option

does not consider their support. Support of peripheral devices should be treated by a separate option.)

The screen of the data entry terminal is a rectangle M characters by N lines. The values of M and N are set by negotiating the Output Line Width and Output Page Size options, respectively. The next writing position (x,y) on the screen (where x is the character position and y is the position of the line on the screen) is indicated by a special display character called the cursor. The cursor may be moved to any position on the screen without disturbing any characters already on the screen. Cursor addressing in existing terminals utilizes several topologies and addressing methods. In order to make the burden of implementaton as easy as possible this protocol supports two topologies (the finite plane and the helical torus) and three addressing methods ((x,y); x and y, and relative increments). Since the finite plane with absolute addressing is the least ambiguous and the easiest to translate to and from the others, it is the default scheme used by the NVDET. The torodial form with either relative or absolute addressing is provided for convience.

Also the NVDET provides a mechanism for defining on the screen fields with special attributes. For example, characters entered into these fields may be displayed with brighter intensity, highlighted by reverse video or blinking, or protected from modification by the user. This latter feature is one of the most heavily used for applications where the DET displays a form to be filled out by the user.

The definition of the NVDET uses Telnet option subnegotiations to accomplish all of its functions. Since none of the ASCII characters sent in the data stream have been used to define these functions, the DET option can be used in a "raw" or even "rare" mode. In circumstances where the application program knows what kind of terminal is on the other end, it can send the ASCII characters required to control functions not supported by the option or an implementation. In general keeping all NVDET functions out of the data stream provides better flexibility.

Facility Functions (for detailed semantics see Section 5.)

IAC SB DET <DET facility subcommand><facility map> IAC SE

where <DET facility subcommand> is one 8-bit byte

indicating the class of the facilities to be described, and <facility map> is a field of one or two 8-bit bytes containing flags describing the facilities required or desired by the sender. The bits of the facility maps are numbered from the right starting at zero. Thus, if bit 2 is set the field will have a decimal value of 4. The values of the field are as follows:

facility cmd: EDIT FACILITIES subcommand code: 1

facility map:	bit numbers
Toroidal Cursor Addressing	6
Incremental Cursor Addressing	5
Read Cursor Address	4
Line Insert/Delete	3
Char Insert/Delete	2
Back Tab	1
Positive Addressing only	0

where:

If the Toroidal Cursor Addressing bit is set, the sender requests or provides that the SKIP TO LINE and SKIP TO CHAR subcommands be supported.

If the Incremental Cursor Addressing bit is set, the sender requests or provides that the UP, DOWN, LEFT, and RIGHT subcommands be supported.

If the Read Cursor bit is set, the sender requests or provides the READ CURSOR subcommand.

If the Line Insert/Delete bit is set, the sender requests or provides that the LINE INSERT and LINE DELETE subcommands be supported.

If the Char Insert/Delete bit is set, the sender requests or provides that the CHAR INSERT and CHAR DELETE subcommands be supported.

If the Back Tab bit is set, the sender requests or provides that the BACK TAB subcommand be supported.

If the Positive Addressing bit is set, then the sender is informing the receiver that it can only move the cursor in the positive direction. (Note: Terminals that have this property also have a Home function to get back to the beginning.)

facility cmd: ERASE FACILITIES subcommand code: 2

facility map:	bit numbers

Erase Field	4
Erase Line	3
Erase Rest of Screen	2
Erase Rest of Line	1
Erase Rest of Field	0

where:

If a bit of the facility map for this facility command is set, the sender requests or provides the facility indicated by the bit. For a more complete description of each of these functions see the Erase Functions section below.

facility cmd: TRANSMIT FACILITIES subcommand code: 3

facility map:	bit numbers
Data Transmit	5
Transmit Line	4
Transmit Field	3
Transmit Rest of Screen	2
Transmit Rest of Line	1
Transmit Rest of Field	0

where:

If a bit of the facility map for this facility command is set, the sender requests or provides the facility indicated by the bit. For a more complete description of each of these functions see the Transmit Functions section below.

facility cmd: FORMAT FACILITIES subcommand code: 4

facility map:	bit numbers
byte 0	
Repeat	4
Blinking	3
Reverse Video	2
Right Justification	1
Overstrike	0
byte 1	
Protection On/Off	6
Protection	5
Alphabetic-only Protection	4
Numeric-only Protection	3
Intensity	0-2

where:

If the Repeat bit is set the sender requests or provides

the REPEAT subcommand.

If the Blinking bit is set, the sender requests or provides the ability to highlight a string of characters by causing them to blink.

If the Reverse Video bit is set, the sender requests or provides the ability to highlight a string of characters by "reversing the video image," i.e., if the characters are normally displayed as black characters on a white background, this is reversed to be white characters on a black background, or vice versa.

If the Right Justification bit is set, the sender requests or provides the ability to cause entries of data to be right justified in the field.

If the Overstrike bit is set, the sender requests or provides the ability to superimpose one character over another on the screen much like a hard copy terminal would do if the print mechanism struck the same position on the paper with different characters.

If the Protection On/Off bit is set, the sender requests or provides the ability to turn on and off field protection.

If the Protection bit is set, the sender requests or provides the ability to protect certain strings of characters displayed on the screen from being altered by the user of the terminal. Setting this bit also implies that ERASE UNPROTECTED and TRANSMIT UNPROTECTED subcommands (see below) are supported.

If the Alphabetic-only Protection bit is set, the sender requests or provides the ability to constrain the user of the terminal such that he may only enter alphabetic data into certain areas of the screen.

If the Numeric-only Protection bit is set, the sender requests or provides the ability to constrain the user of the terminal such that he may only enter numerical data into certain areas of the screen.

The three bits of the Intensity field will contain a positive binary integer indicating the number of levels of intensity that the sender requests or provides for displaying the data. The value of the 3 bit field should be interpreted in the following way:

1	one visible intensity
2	two intensities; normal and bright
3	three intensities; off, normal, and bright
>3	>3 intensities; off, and the remaining levels proportioned from dimmest to brightest intensity.

For the all of the above commands, if the appropriate bit in <facility map> is not set, then the sender does not request or provide that facility.

Editing Functions

IAC SB DET MOVE CURSOR <x><y> IAC SE subcommand code: 5

where <x> is an 8-bit byte containing a positive binary integer representing the character position of the cursor, <y> is an 8-bit byte containing a positive binary integer representing the line position of the cursor.

This subcommand moves the cursor to the absolute screen address (x,y) with the following boundary conditions:

if x>M-1, set x=M-1 and send an ERROR subcommand
if y>N-1, set y=N-1 and send an ERROR subcommand

This describes a finite plane topology on the screen.

IAC SB DET SKIP TO LINE <y> IAC SE subcommand code: 6

where <y> is a positive 8-bit binary number.

This subcommand moves the cursor to the absolute screen line y. x remains constant. For values of y>N-1
y = y mod N.

IAC SB DET SKIP TO CHAR <x> IAC SE subcommand code: 7

where <x> is a positive 8-bit binary number.

This subcommand moves the cursor to the absolute character position x. y remains constant, unless x>M-1 in which case:
x' = (x mod M)
y' = (y+(x DIV N))
where x' and y' are the new values of the cursor.

These last two subcommands define a toroidal topology on the screen.

IAC SB DET UP IAC SE subcommand code: 8
IAC SB DET DOWN IAC SE subcommand code: 9
IAC SB DET LEFT IAC SE subcommand code: 10
IAC SB DET RIGHT IAC SE subcommand code: 11

These subcommands are provided as a convenience for some terminals. The commands UP, DOWN, LEFT, and RIGHT are defined as

UP: (x,y)=(x, y-1 mod N)
DOWN: (x,y)=(x, y+1 mod N)
LEFT: (x,y)=(x-1, y); if x=0 then x-1 = 0
RIGHT: (x,y)=(x+1 mod M, y) and y = y+1 if x+1>M-1

Note: DOWN, LEFT, and RIGHT cannot always be replaced by the ASCII codes for linefeed, backspace, and space respectively. The latter are format effectors while the former are cursor controls.

IAC SB DET HOME IAC SE subcommand code: 12

This subcommand positions the cursor to (0,0). This is equivalent to a MOVE CURSOR 0,0 or the sequence SKIP TO LINE 0, SKIP TO CHAR 0. This subcommand is provided for convenience, since most terminals have\177 it as a separate control.

IAC SB DET LINE INSERT IAC SE subcommand code: 13

This subcommand inserts a line of spaces between lines y (the current line, determined by the position of the cursor) and line y-1. Lines y through N-2 move down one line, i.e. line y becomes line y+1; y+1 becomes y+2, ...; N-2 becomes N-1. Line N-1 is lost off the bottom of the screen. The position of the cursor remains unchanged.

IAC SB DET LINE DELETE IAC SE subcommand code: 14

This subcommand deletes line y where y is the current line position of the cursor. Lines y+1 through N-1 move up one line, i.e. line y+1 becomes line y; y+2 becomes y+1; ...; N-1 becomes N-2. The N-1st line position is set to all spaces. The cursor position remains unchanged.

IAC SB DET CHAR INSERT IAC SE subcommand code: 15

This subcommand inserts the next character in the data stream between the xth and x-1st characters, where x is

the current character position of the cursor. The xth through M-2nd characters on the line are shifted one character positon to the right. The new character is inserted at the vacated xth position. The M-1st character is lost. The position of the cursor remains unchanged.

IAC SB DET CHAR DELETE IAC SE subcommand code: 16

This subcommand deletes the character on the screen at the x-th position. The x-th character is removed and the characters x+1 through M-1 are shifted one character position to the left to become the x-th through M-2nd characters. The M-1st character position is left empty. (For most terminals it will be set to a NUL or space.) The cursor position remains unchanged.

IAC SB DET READ CURSOR IAC SE subcommand code: 17

This subcommand requests the receiver to send the present position of the cursor to the sender.

IAC SB DET CURSOR POSITION <x><y> IAC SE
subcommand code: 18

where <x> and <y> are positive 8-bit binary integers.

This subcommand is sent by a Telnet implementation in response to a READ CURSOR subcommand to convey the coordinates of the cursor to the other side. Note: x is less than M and y is less than N.

IAC SB DET REVERSE TAB IAC SE subcommand code: 19

This subcommand causes the cursor to move to the previous tab position. If none exists on the present line, the cursor moves to the previous line and so on until a tab is found or the address (0,0) is encountered. When field protection is in effect the cursor moves to the beginning of the preceding unprotected field.

Transmit Functions (For detailed semantics see Section 5.)

IAC SB DET TRANSMIT SCREEN IAC SE subcommand code: 20

This subcommand causes the terminal to transmit all characters on the screen from position (0,0) to (M-1,N-1). The cursor will be at (0,0) after the operation is complete.

IAC SB DET TRANSMIT UNPROTECTED IAC SE
subcommand code: 21

This subcommand causes the terminal to transmit all characters in unprotected fields from position (0,0) to (M-1,N-1). The unprotected fields are separated by the field separator subcommand. The cursor will be at (0,0) or at the beginning of the first unprotected field after the operation is complete.

IAC SB DET TRANSMIT LINE IAC SE subcommand code: 22

This subcommand causes the terminal to transmit all data on the yth line where y is determined by the present position of the cursor. Data is sent from character position (0,y) to the end-of-line or position (M-1,y) whichever comes first. The cursor position after the transmission is one character position after the end of line condition or the beginning of the next line, (0,y+1).

IAC SB DET TRANSMIT FIELD IAC SE subcommand code: 23

This subcommand causes the terminal to transmit all data in the field presently occupied by the cursor. The cursor position after the operation is complete is one character position after the end of the field or, if that position is protected, at the beginning of the next unprotected field.

IAC SB DET TRANSMIT REST OF SCREEN IAC SE
subcommand code: 24

This subcommand causes the terminal to transmit all characters on the screen from position (x,y) to (M-1,N-1) or until the end of text. (x,y) is the current cursor position. The cursor position after the operation is one character position after the last text character, or (0,0) if the last filled character position is (M-1,N-1).

IAC SB DET TRANSMIT REST OF LINE IAC SE
subcommand code: 25

This subcommand causes the terminal to transmit all characters on the yth line from position (x,y) to the end of line or (M-1,y) whichever comes first. (x,y) is the current cursor position. The cursor position after the operation is one character position after the last character of the line or the first character of the next line.

IAC SB DET TRANSMIT REST OF FIELD IAC SE
subcommand code: 26

This subcommand causes the receiver to transmit the rest of the characters in the field currently occupied by the cursor. The cursor position after the operation is at the beginning of the next field.

IAC SB DET DATA TRANSMIT <x><y> IAC SE
subcommand code: 27

This subcommand is used to preface data sent from the terminal in response to a user action or a TRANSMIT command. The parameters <x> and <y> indicate the initial position of the cursor. See the Transmit Subcommands subsection in Section 5 for more details.

Erase Functions

IAC SB DET ERASE SCREEN IAC SE subcommand code: 28

This subcommand causes all characters to be removed from the screen. All fields regardless of their attributes are deleted. The cursor position after the operation will be (0,0). Most terminals set the erased characters to either NUL or space characters.

IAC SB DET ERASE LINE IAC SE subcommand code: 29

This subcommand causes all characters on the yth line to be removed from the screen, where y is the line of the current cursor position. All fields regardless of their attributes are deleted. The cursor position after this operation will be (0,y). Note: This operation can be easily simulated by the sequence: LINE DELETE, LINE INSERT. However, the order is important to insure that no data is lost off the bottom of the screen.

IAC SB DET ERASE FIELD IAC SE subcommand code: 30

This subcommand causes all characters in the field occupied by the cursor to be removed. The cursor position after the operation is at the beginning of the field.

IAC SB DET ERASE REST OF SCREEN IAC SE
subcommand code: 31

This subcommand causes all characters from position (x,y) to (M-1,N-1) to be removed from the screen. All fields regardless of their attributes are deleted. The cursor position after the operation is unchanged. This is equivalent to doing an ERASE REST OF LINE plus a LINE DELETE for lines greater than y.

IAC SB DET ERASE REST OF LINE IAC SE
subcommand code: 32

This subcommand causes all characters from position (x,y) to (M-1,y) to be removed from the screen All fields regardless of their attributes are deleted. The cursor position after the operation is unchanged.

IAC SB DET ERASE REST OF FIELD IAC SE
subcommand code: 33

This subcommand causes all characters from position (x,y) to the end of the current field to be removed from the screen. The cursor position after the operation is unchanged.

IAC SB DET ERASE UNPROTECTED IAC SE
subcommand code: 34

This subcommand causes all characters on the screen in unprotected fields to be removed from the screen. The cursor position after the operation is at (0,0) or, if that position is protected, at the beginning of the first unprotected field.

Format Functions

IAC SB DET FORMAT DATA <format map><count> IAC SE
subcommand code: 35

where <format map> is an 8-bit byte containing the following flags:

Blinking	7
Reverse Video	6
Right Justification	5
Protection	3-4
Intensity	0-2

where:

If the Blinking bit is set, the following field of <count> characters should have the Blinking attribute

applied to it by the receiver.

If the Reverse Video bit is set, the following field of <count> characters should be displayed by the receiver with video reversed.

If the Right Justification bit is set, the input entered into the field of <count> characters should be right justified.

The Protection field is two bits wide and may take on the following values:

0	no protection
1	protected
2	alphabetic only
3	numeric only

The protection attribute specifies that the other side may modify any character (no protection), modify no characters (protected), enter only alphabetical characters (A-Z, and a-z) (alphabetic only), or enter only numerical characters (0-9,+,.,and -) (numeric only) in the following field of <count> bytes.

The Intensity field is 3 bits wide and should be interpreted in the following way:

The values 0-6 should be used as an indication of the relative brightness to be used when displaying the characters in or entered into the following field <count> characters wide. The number of levels of brightness available should have been obtained previously by the Format Facility subcommand. The exact algorithm for mapping these values to the available levels of intensity is left to the implementors. A value of 7 in the intensity field indicates that the brightness should be off, and any characters in or entered into the field should not be displayed.

<count> is 2 bytes that should be interpreted as a positive 16-bit binary integer representing the number of characters following this command which are affected by it.

Data sent to the terminal or the Using Host for unwritten areas of the screen not in the scope of the count should be displayed with the default values of the format map. The default values are No Blinking, Normal Video, No Justification, No Protection and Normal Intensity.

This subcommand is used to format data to be displayed on the screen of the terminal. The <format map> describes the attributes that the field <count> bytes wide should have. This field is to start at the position of the cursor when the command is acted upon. The next <count> displayable characters in the data stream are used to fill the field. Subsequent REPEAT subcommands may be used to specify the contents of this field. If the sender specifies attributes that have not been agreed upon by the use of the Format Facility subcommand, the Telnet process should send an Error Subcommand to the sender, but format the screen as if the bit had not been set.

IAC SB DET REPEAT <count><char> IAC SE
subcommand code: 36

where <count> is a positive 8-bit binary integer. <char> is an 8-bit byte containing an ASCII character.

This subcommand is used to perform data compression on data being transferred to the terminal by encoding strings of identical characters as the character and a count. The repeated characters may be part of a field specified\177\177

IAC SB DET SUPPRESS PROTECTION <negotiation> IAC SE
subcommand code: 37

where <negotiation> may have the values of the Telnet option negotiation:

251	WILL
252	WONT
253	DO
254	DONT

This subcommand is used to suppress the field protection in a non-destructive manner. Many data entry terminals provide the means by which protection may be turned on and off without modifying the contents of the screen or the terminal’s memory. Thus, the protection may be turned off and back on without retransmitting the form. The default setting of the option is that protection is on, in other words

IAC SB DET SUPPRESS PROTECTION WONT IAC SE
IAC SB DET SUPPRESS PROTECTION DONT IAC SE

Negotiation of this subcommand follows the same rules as negotiations of the Telnet options.

IAC SB DET FIELD SEPARATOR IAC SE subcommand code: 38

It is necessary when transmitting only the unprotected portion of the screen to provide a means for delimiting the fields. Existing DET's use a variety of ASCII characters such as Tab, Group Separator, Unit Separator, etc. In order to maintain transparency of the NVDET this subcommand is used to separate the fields. Clearly, this incurs rather high overhead. This overhead can be avoided by using the Byte Macro Option (see Appendix 3).

Miscellaneous Commands

IAC SB DET FN <code> IAC SE subcommand code: 39

where: <code> is one byte.

Many data-entry terminals provide a set of "function" keys which when pressed send a one-character command to the server. This subcommand describes such a facility. The values of the <code> field are defined by the user and server. The option merely provides the means to transfer the information.

IAC SB DET ERROR <cmd> <error code> IAC SE
subcommand code: 40

where:
<cmd> is a byte containing the subcommand code of the subcommand in error.
<error code> is a byte containing an error code. (For a list of the defined error codes see Appendix 2.)

This subcommand is provided to allow DET option implementations to report errors they detect to the corresponding Telnet process. At this point it is worth reiterating that the philosophy of this option is that when an error is detected it should be reported; however, the implementation should attempt its best effort to carry out the intent of the subcommand or data in error.

3. Default and Minimal Implementation Specifications

Default

WON'T DET -- DON'T DET

Neither host wishes to use the Data Entry Terminal option.

Minimal Implementation

DET EDIT FACILITIES
DET ERASE FACILITIES
DET TRANSMIT FACILITIES
DET FORMAT FACILITIES
DET MOVE CURSOR <x><y>
DET HOME
DET ERASE SCREEN
DET TRANSMIT SCREEN
DET FORMAT DATA
DET ERROR <cmd> <error code>

In the case of formatting the data, the minimal implementation should be able to support a low and high level of intensity and protection for all or no characters in a field. These functions, however, are not required.

The minimal implementation also requires that the Output Line Width and Output Page Size Telnet options be supported.

4. Motivation

The Telnet protocol was originally designed to provide a means for scroll-mode terminals, such as the standard teletype, to communicate with processes through the network. This was suitable for the vast majority of terminals and users at that time. However, as use of the network has increased into other areas, especially areas where the network is considered to provide a production environment for other work, the desires and requirements of the user community have changed. Therefore, it is necessary to consider supporting facilities that were not initially supported. This Telnet option attempts to do that for applications that require data entry terminals.

This option in effect defines the Network Virtual Data Entry Terminal. Although the description of this option is quite long, this does not imply that the Telnet protocol is a poor vehicle for this facility. Data Entry Terminals are rather complex and varied in their abilities. This option attempts to support both the minimal set of useful functions that are either common to all or can be easily simulated and the more sophisticated functions supplied in some terminals.

Unlike most real data entry terminals where the terminal functions are encoded into one or more characters of the native character set, this option performs all such controls within the Telnet subnegotiation mechanism. This allows programs that are intimately familiar with the kind of terminal they are communicating with to send commands that may not be supported by either the option or the implementation. In other words, it is possible to operate in a "raw" or at least "rare" mode using as much of the option as necessary.

Although many data entry terminals support a variety of peripheral devices such as printers, cassettes, etc. it is beyond the scope of this option to entertain such considerations. A separate option should be defined to handle this aspect of these devices.

5. Description

General Notes

All implementations of this option are required to support a certain minimal set of the subcommands for this option. Section 3 contains a complete list of the subcommands in this minimal set. In keeping with the Telnet protocol philosophy that an implementation should not have to be able to parse commands it does not implement, every subcommand of this option is either in the minimal set or is covered by one of the facility subcommands. An implementation must "negotiate" with its correspondent for permission to use subcommands not in the minimal set before using them. For details of this negotiation process see the section below on facility subcommands.

Most data entry terminals are used in a half duplex mode. (Although most DET's on the market can be used either as data entry terminals or as standard interactive terminals, we are only concerned here with their use as DET's.) When this option is used, it is suggested that the following Telnet options be refused: Echo, Remote Controlled Transmission and Echoing, and Suppress Go-Ahead. However, this option could be used to support a simple full duplex CRT based application using the basic cursor control functions provided here. For these cases, one or more of the above list of options might be required. (Support of sophisticated interactive calligraphic applications is beyond the scope of this option and should be done by another option or the Network Graphics Protocol.)

In RFC 728, it was noted that a synch sequence can cause undesired interactions between Telnet Control functions and the data stream. A synch sequence causes data but not control functions to be flushed. If a control function which has an effect on the data immediately following it is present in the data stream when a synch sequence occurs, the control function will have its effect not on the intended data but on the data immediately following the Data Mark. The following DET subcommands are susceptible to this pitfall:

CHAR INSERT
DATA TRANSMIT
FORMAT DATA

The undesired interactions are best avoided by the receiver

of the synch sequence deleting these subcommands\177 and all data associated with them before continuing to process the control functions. This implies that the Data Mark should not occur in the middle of the data associated with these subcommands.

Facility Subcommands

These four subcommands are used by the User and Server implementations to negotiate the subcommands and attributes of the terminal that may be utilized. This negotiation can be viewed as the terminal (User Host) indicating what facilities are provided and the Server Host (or application program) indicating what facilities are desired.

When Sent: A Server Telnet implementation using the DET option must send a facility subcommand requesting the use of a particular subcommand or terminal attribute not in the minimal implementation before the first use of that subcommand or attribute. The User Telnet implementation should respond as quickly as possible with its reply. Neither the User nor Server are required to negotiate one subcommand at a time. Also, a Telnet implementation responding to a facility subcommand is not required to give permission only for that subcommand. It may send a format map indicating all facilities of that class which it supports. However, a Telnet implementation requesting facilities must send a facility subcommand before its first use of the subcommand regardless of whether earlier negotiations have indicated the facility is provided. The facility cannot be used until a corresponding facility subcommand has been received. There are no other constraints on when the facility subcommands may be sent. In particular, it is not necessary for an application to know at the beginning of a session all facilities that it will use.

Action When Recieved: There are two possible actions that may be taken when a facility subcommand is received depending on whether the receiver is a requestor or a provider (User).

Requestor: When a facility subcommand is received by a requestor and it is in the state of Waiting for a Reply, it should go into the state of Not Waiting. It should then take the facility map it had sent and form the logical intersection with the facility map received. (For the Intensity attribute, one should take the minimum of the number received and the number requested.) The result indicates the facilities successfully negotiated. Note: if

the receiver is not in the Waiting for Reply state, then this is the provider case described next.

Provider: When a facility subcommand is received, it should send a facility subcommand with a facility map of the facilities it provides as soon as possible. It should then determine what new facilities it is providing for the Requestor by forming the logical intersection of the facility map received and the one sent.

Note: Although in most cases the requestor will be the Server Host and the provider will be the User Host supporting the terminal, this distinction may not always be true.

Transmit Subcommands

There are two kinds of transmit subcommands: those used to request that data be sent to the requestor, and one to preface data sent to the requestor. The first kind allow the requestor to control when, from where and to some degree how much data is transmitted from the terminal. Their explanation is straightforward and may be found in Section 2.

Data may be sent from the terminal as a result of two events: the user of the terminal caused the transmission or in response to a transmit subcommand. Some programs may wish to know from where on the screen the transmission began. (This is reasonable, since the terminal user may move the cursor around considerably before transmitting.) Other programs may not need such information. The DATA TRANSMIT subcommand is provided in case this function is needed. When used this subcommand prefaces data coming from the terminal. The parameters <x> and <y> give the screen coordinates of the beginning of the transmission. <x> must be less than or equal to M-1 and <y> must be less than or equal to N-1. It is assumed that all data between this DATA TRANSMIT and the next one starts at the coordinates given by the first subcommand and continues filling each line thereafter according to the constraints of the screen and the format effectors in the data. Thus an intelligent or sloppy user-host DET implementation (depending on your point of view) need only include a DATA TRANSMIT subcommand when the new starting point is different from the last ending point.

6. Sample Interaction

The nomenclature of RFC 726 will be used to describe this example. To quote that RFC:

"S:" is sent from serving host to using host.
"U:" is sent from using host to serving host.
"T:" is entered by the terminal user.
"P:" is printed on the terminal.

Text surrounded by square brackets([]) is commentary. Text surrounded by angle brackets (<>) is to be taken as a single unit. E.g, carriage return is <cr>, and the decimal value 27 is represented <27>.

We assume that the user has established the Telnet connection, logged on, and an application program has just been started either by the user directly or through a canned start up procedure. The presentation on the page is meant to merely group entities together and does not imply the position of message boundaries. One should assume that any part of the dialogue may be sent as one or many messages. The first action of the program or Telnet is to negotiate the DET option:

S: <IAC><DO><DET>

U: <IAC><WILL><DET>

S:<IAC><DO><OUTPUT PAGE SIZE>
U:<IAC><WILL><NAOP>
U:<IAC><SB><NAOP><DR><25><IAC><SE>
S:<IAC><SB><NAOP><DS><0><IAC><SE>

[First negotiate the screen size. In this case we are asking the user the size of the terminal. This could have been done before the DET option was negotiated.]

S:<IAC><DO><OUTPUT LINE WIDTH>

U:<IAC><SB><NAOL><DR><80><IAC><SE>
S:<IAC><SB><NAOL><DS><0><IAC><SE>

[Defines the screen to be 25 lines by 80 characters. The server may use this information when formatting the screen.]

S:<IAC><SB><DET><FORMAT FACILITIES> <Repeat><Protection, 3 Levels Intensity><IAC><SE>

[Now set the terminal attributes.]

U:<IAC><SB><DET><FORMAT FACILITIES> <Repeat, Blinking><Protection, 3 Levels Intensity><IAC><SE>

S:<IAC><SB><DET><ERASE SCREEN><IAC><SE>

[Erase the screen and

start sending the form.]

```
  <IAC><SB><DET><FORMAT DATA>
  <Protection=1, Intensity=1><0>
  <5><IAC><SE>Name:

  <IAC><SB><DET><MOVE CURSOR><0><1>
    <IAC><SE>

  <IAC><SB><DET><FORMAT DATA>
    Protection=1, Intensity=1><0>
    <8><IAC><SE>

  Address:
  <IAC><SB><MOVE CURSOR><0><4><IAC>
    <SE>

  <IAC><SB><DET><FORMAT DATA>
    Protection=1, Intensity=1><0>
    <17><IAC><SE>
  Telephone number:

  <IAC><SB><DET><MOVE CURSOR><32><4>
  <IAC><SE>

  <IAC><SB><DET><FORMAT DATA>
    Protection=1, Intensity=1><0>
    <24><IAC><SE>
  Social Security Number:

  <IAC><SB><DET><FORMAT DATA>          [Establish a field that doesn't
    <Protection=1, Intensity=7>        display what is typed into it.]
    <0><11><IAC><SE>

  <IAC><SB><DET><MOVE CURSOR><32>
    <5><IAC><SE>

  <IAC><SB><DET><FORMAT FACILITIES>    [Get permission to use Blinking
    <Blinking><0><IAC><SE>             Attribute.]

U:<IAC><SB><DET><FORMAT FACILITIES>
    <Repeat, Blinking><Protection,
    3 Levels Intensity><IAC><SE>

S:<IAC><SB><DET><FORMAT DATA>
  <Blinking=1, Protection=1,
  Intensity=1><0><29><IAC><SE>

  Your SSN will not be printed.

  <IAC><SB><DET><HOME><IAC><SE>
```

<IAC><GA>

The previous exchange has placed a form on the screen that looks like:

Name:
Address:

Telephone Number: Social Security Number:
"Your SSN will not be printed."

where the quoted string is blinking.

The terminal user is now free to fill in the form provided. He positions the cursor at the beginning of the first field (this usually is done by hitting the tab key) and begins typing. We do not show this interaction since it does not generate any interaction with the User Telnet program or the network. After the terminal user has completed filling in the form, he strikes the transmit key to send the unprotected part of the form, but first the User Telnet program negotiates the Byte Macro Option to condense the Field Separator subcommand:

U:<IAC><DO><BM> [Negotiate Byte Macro Option.]

S:<IAC><WILL><BM> [Define decimal 166 to be the Field Separator subcommand
U:<IAC><SB><BM><DEFINE> (see Appendix 3)]
<166><6><IAC SB DET FIELD
SEPARATOR IAC SE><IAC><SE>

S:<IAC><SB><BM><ACCEPT><166> [The server accepts the macro.]
<IAC><SE>

U:<IAC><SB><DET><DATA TRANSMIT><0><6>
<IAC><SE>
John Doe <166> 1515 Elm St., Urbana, Il 61801
<166> 217-333-9999 <166> 123-45-6789 <166>

S:<IAC><SB><DET><ERASE SCREEN><IAC><SE>
Thank you.

And so on.

Appendix 1 - Subcommands, opcodes and syntax

1	EDIT FACILITIES	<Facilty map>
2	ERASE FACILITIES	<Facility map>
3	TRANSMIT FACILITIES	<Facility map>
4	FORMAT FACILITIES	<Facility map 1> <Facility map 2>
5	MOVE CURSOR	<x> <y>
6	SKIP TO LINE	<y>
7	SKIP TO CHAR	<x>
8	UP	
9	DOWN	
10	LEFT	
11	RIGHT	
12	HOME	
13	LINE INSERT	
14	LINE DELETE	
15	CHAR INSERT	
16	CHAR DELETE	
17	READ CURSOR	
18	CURSOR POSITION	<x><y>
19	REVERSE TAB	
20	TRANSMIT SCREEN	
21	TRANSMIT UNPROTECTED	
22	TRANSMIT LINE	
23	TRANSMIT FIELD	
24	TRANSMIT REST OF SCREEN	
25	TRANSMIT REST OF LINE	
26	TRANSMIT REST OF FIELD	
27	DATA TRANSMIT <x><y>	
28	ERASE SCREEN	
29	ERASE LINE	
30	ERASE FIELD	
31	ERASE REST OF SCREEN	
32	ERASE REST OF LINE	
33	ERASE REST OF FIELD	
34	ERASE UNPROTECTED	
35	FORMAT DATA <format map>	
36	REPEAT	<count><char>
37	SUPPRESS PROTECTION <negotiation>	
38	FIELD SEPARATOR	
39	FN <code>	
40	ERROR	<cmd><error code>

Appendix 2 - Error Codes

1	Facility not previously negotiated.
2	Illegal subcommand code.
3	Cursor Address Out of Bounds.
4	Undefined FN value.
4	Can’t negotiate acceptable line width.
5	Can’t negotiate acceptable page length.
6	Illegal parameter in subcommand.
7	Syntax error in parsing subcommand.
8	Too many parameters in subcommand.
9	Too few parameters in subcommand.
10	Undefined parameter value
11	Unsupported combination of Format Attributes

Appendix 3 - Use of the Byte Macro Option

One of the major drawbacks of the DET option is that because the functions are encoded as Telnet option subnegotiations a fairly high overhead is incurred. A function like Character Insert which is encoded as a single byte in most terminals requires six bytes in the DET option. Originally the only other solution that would have accomplished the same transparency that the use of subcommands provides would have been to define additional Telnet control functions. However, since this would entail modification of the Telnet protocol itself, it was felt that this was not a wise solution. Since then the Telnet Byte Macro Option (RFC 729) has been defined. This option allows the user and server Telnets to map an arbitrary character string into a single byte which is then transferred over the net. Thus the Byte Macro Option provides the means for implementations to avoid the overhead for heavily used subcommands. The rest of this appendix suggests how the Byte Macro Option should be applied to the DET option.

In keeping with the specification of the Byte Macro Option, macro bytes will be chosen from the range 128 to 239. For the DET option, it is suggested that macro bytes be chosen by adding the subcommand code to 128. In addition, an unofficial DET subcommand might be defined indicating that each side was willing to support macro bytes for all subcommands (but not necessarily support all of the subcommands themselves) according to this algorithm. This subcommand would be:

```
IAC SB DET DET-MACRO <negotiation> IAC SE
                                   subcommand code:  254
```

where <negotiation> may have the values of the Telnet option negotiation:

```
              251       WILL
              252       WONT
              253       DO
              254       DONT
```

This subcommand is sent by a Telnet implementation to indicate its willingness to adopt byte macros for all of the DET subcommands according to the following algorithm:

The macro byte for subcommand i will be i+128 and will represent the following string for parameterless subcommands:

IAC SB DET <subcommand code> IAC SE

and the following string for subcommands with parameters:

IAC SB DET <subcommand code>

The default setting for this subcommand is that the macros are not in effect, in other words,

```
IAC SB DET DET-MACRO WONT IAC SE
IAC SB DET DET-MACRO DONT IAC SE
```

Negotiation of this subcommand follows the same rules as negotiations of the Telnet options.

References

1. ADM-1 Interactive Display Terminal Operator's Handbook
Lear-Siegler, Inc. 7410-31.

2. ADM-Interactive Display Terminal Operator's Handbook
Lear-Siegler, Inc. EID, 1974.

3. Burroughs TD 700/800 Reference Manual, Burroughs Corp., 1973

4. Burroughs TD 820 Reference Manual, Burroughs Corp. 1975.

5. CC-40 Communications Station: General Information Manual.
Computer Communication, Inc. Pub. No. MI-1100. 1974.

6. Crocker, David. "Telnet Byte Macro Option," RFC 729, 1977.

7. Data Entry Virtual Terminal Protocol for Euronet, DRAFT, 1977.

8. Day, John. "A Minor Pitfall in the Telnet Protocol,"
RFC 728, 1977.

9. Hazeltine 2000 Desk Top Display Operating Instructions.
Hazeltine IB-1866A, 1870.

10. How to Use the Consul 980: A Terminal Operator's Guide
and Interface Manual. Applied Digital Data Systems, Inc.
98-3000.

11. How to Use the Consul 520: A Terminal Operator's Guide
and Interface Manual. Applied Digital Data Systems, Inc.
52-3000.

12. Honeywell 7700 Series Visual Information Projection (VIP)
Systems: Preliminary Edition. 1973.

13. An Introduction to the IBM 3270 Information Display System.
IBM GA27-2739-4. 1973.

14. Naffah, N. "Protocole Appareil Virtuel type Ecran"
Reseau Cyclades. TER 536. 1976.

15. Postel, Jon and Crocker, David. "Remote Controlled Transmission and Echoing Telnet Option", RFC 726 NIC 39237, Mar. 1977.

16. Schicker, Peter. "Virtual Terminal Protocol (Proposal 2). INWG Protocol Note #32., 1976.

17. UNISCOPE Display Terminal : Programmer Reference . Sperry-Univac UP-7807 Rev. 2, 1975.

18. Universal Terminal System 400: System Description. Sperry-Univac UP-8357, 1976.

19. Walden, David C. "Telnet Output Line Width Option." NIC # 20196, 1973, also in ARPANET Protocol Handbook, 1976.

20. Walden, David C. "Telnet Output Page Size" NIC # 20197, 1973, also in ARPANET Protocol Handbook, 1976.