Network Working Group A. Yasuda
Request for Comments: 1043 T. Thompson
Defense Intelligence Agency
Updates: RFC 732 February 1988

TELNET Data Entry Terminal Option
DODIIS Implementation

Status of this Memo

This RFC suggests a proposed protocol on the TELNET Data Entry Terminal (DET) Option - DODIIS Implementation for the Internet community. It is intended that this specification be compatible with the specification of DET Option in RFC-732. Discussion and suggestions for improvements are encouraged. Distribution of this memo is unlimited.

Introduction

In the early 1980s, the Defense Intelligence Agency (DIA) undertook the tasks of developing a TELNET capability to access full screen applications across a packet switching network. This effort was successful by implementing Data Entry Terminal (DET) options within the TELNET protocol based on RFC 732. These DET options have been implemented on IAS, MVS, OS86 and UNIX operating systems. DET options are being developed for VM and VMS operating systems.

The Department of Defense Intelligence Information System (DODIIS) is a confederation of heterogeneous computer systems and remote terminals utilizing the Defense Data Network (DDN) as the communications backbone (namely the SCINET/DSNET-3).

Although the reason for implementing a DET option specification was based upon data base application interfaces, the use of a full screen TELNET provides a method to achieve higher efficiency on the network. Most terminal to host applications on the ARPANET are character echo TELNETs. This is both costly in time and network utilization, since one character pressed on the keyboard generates a datagram composed of TCP/IP headers plus the character sent to the host and the host echoes back a similar datagram. In the DODIIS community, programmers are highly encouraged to implement full screen applications; line at a time is acceptable; and character remote echo mode is discouraged.

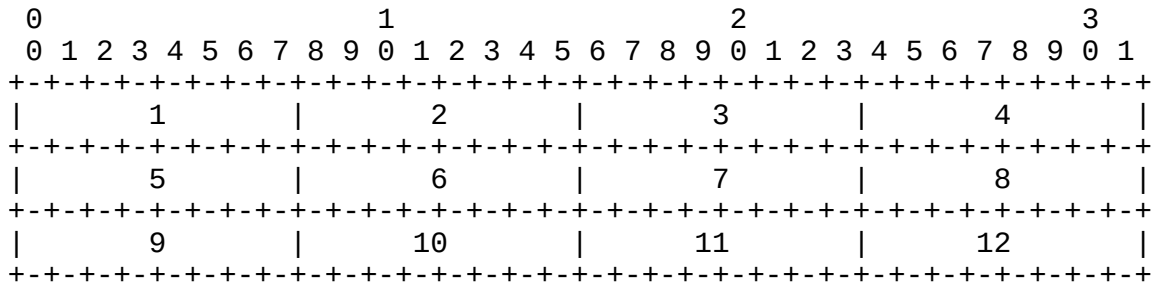
This RFC in its final form will be implemented on SCINET. During the interim period, the "DODIIS TELNET Network Virtual Data Entry Terminal (NVDET) Option Specification", DIA, April 1983, will be implemented.

TABLE OF CONTENTS

		Page No.
SECTION 1	COMMAND NAME AND OPTION CODE	4
SECTION 2	COMMAND MEANINGS	4
	Facilities Subcommands	4
	Edit Subcommands	8
	Transmit Subcommands	8
	Erase Subcommands	10
	Format Subcommands	10
	Miscellaneous Subcommands	13
SECTION 3	DEFAULT AND MINIMAL IMPLEMENTATION	15
SECTION 4	MOTIVATION FOR THE OPTION	17
SECTION 5	DESCRIPTION AND IMPLEMENTATION RULES	17
	The DODIIS DET Model	17
	Negotiating the DET Option	18
	DET Facilities Negotiation	18
	General DET Interaction	19
	Form Construction	20
	Form response	21
	Function Keys	22
	Field Selection	22
	Out-Of-Context Data	23
	Line Discipline	23
	Standard TELNET Control Functions	24
	Other Implementation Notes	24
APPENDIX 1	DET OPCODES AND SUBCOMMAND SYNTAX	25
APPENDIX 2	DET ERROR CODES	26

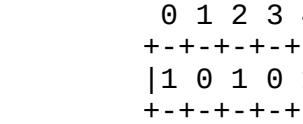
The convention in the documentation of the TELNET NVDET Protocol is to express numbers in decimal. Data fields are described left to right, with the most significant octet on the left and the least significant octet on the right.

The order of transmission of the data described in this document is resolved to the octet level. Whenever a diagram shows a group of octets, the order of transmission of those octets is the normal order in which they are read in English. For example, in the following diagram the octets are transmitted in the order they are numbered.



Transmission Order of Bytes

Whenever an octet represents a numeric quantity, the left most bit in the diagram is the high order or most significant bit. That is, the bit labeled 0 is the most significant bit. For example, the following diagram represents the value 170 (decimal).



Significance of Bits

Similarly, whenever a multi-octet field represents a numeric quantity, the left most bit of the whole field is the most significant bit. When a multi-octet quantity is transmitted the most significant octet is transmitted first.

1. Command Name and Option Code

   DET 20

2. Command Meanings

   IAC WILL DET

      The sender of this command REQUESTS permission to begin, or AGREES that it will begin, sending and receiving Data Entry Terminal (DET) subcommands to control session interactions.

   IAC WONT DET

      If the connection is already operating in DET mode, the sender of this command DEMANDS that the connection stop operating in DET mode and begin operating in TELNET NVT mode. If the connection is not operating in DET mode, the sender REFUSES to begin operating in DET mode. A connection is operating in TELNET NVT mode when both parties are interpreting data as described by the TELNET SPECIFICATION, MIL-STD-1782.

   IAC DO DET

      The sender of this command REQUESTS permission to begin, or AGREES that it will begin, sending and receiving Data Entry Terminal (DET) subcommands to control session interactions.

   IAC DONT DET

      If the connection is already operating in DET mode, the sender of this command DEMANDS that the connection stop operating in DET mode and begin operating in TELNET NVT mode. If the connection is not operating in DET mode, the sender REFUSES to begin operating in DET mode. A connection is operating in TELNET NVT mode when both parties are interpreting data as described by the TELNET SPECIFICATION, MIL-STD-1782.

   DODIIS implementations of the DET option use the subcommands described in the remainder of Section 2. A description of the DODIIS DET model and DET subcommand usage is contained in Section 5.

FACILITIES SUBCOMMANDS. Facilities subcommands are used to negotiate DET facilities (subcommands and attributes). The facility subcommands indicate the DET facilities the sender supports. Facility negotiation may be viewed as the terminal indicating the facilities it provides and the application indicating the facilities

it desires. 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.

IAC SB DET EDIT-FACILITIES <facility map> IAC SE

subcommand code: 1

This subcommand indicates the edit facilities the sender supports. The <facility map> parameter is one eight bit byte containing the following flags:

Bits 5-7	Reserved
Bit 4	Read Cursor
Bits 0-3	Reserved

where:

If the Read-Cursor bit is set, the sender supports the READ-CURSOR and CURSOR-POSITION subcommands.

Reserved bits represent edit facilities that are not defined for DODIIS implementations; therefore, no descriptions are provided. Reserved bits must be zeroed to indicate non support of the associated edit facilities.

IAC SB DET ERASE-FACILITIES <facility map> IAC SE

subcommand code: 2

This subcommand indicates the erase facilities the sender supports. The <facility map> parameter is one eight bit byte containing flags. Since no erase facilities are defined for DODIIS implementations, no descriptions are provided. The ERASE-FACILITIES subcommand is part of the minimal DET implementation and is included for that reason. DODISI implementors must declare non support of erase facilities by sending this subcommand with a zeroed facility map.

IAC SB DET TRANSMIT-FACILITIES <facility map> IAC SE

subcommand code: 3

This subcommand indicates the transmit facilities the sender supports. The <facility map> parameter is one eight bit byte containing the following flags:

```
Bits 6-7        Reserved
Bit  5          Data Transmit
Bits 0-4        Reserved
```

where:

If the Data-Transmit bit is set, the sender supports the DATA-TRANSMIT subcommand.

Reserved bits represent transmit facilities that are not defined for DODIIS implementations; therefore, no descriptions are provided. Reserved bits must be zeroed to indicate non support of the associated transmit facilities.

IAC SB DET FORMAT-FACILITIES <facility map> IAC SE

subcommand code: 4

This subcommand indicates the format facilities the sender supports. The <facility map> parameter is two eight bit bytes containing the following:

```
Byte 0
   Bit  7       Function Key
   Bit  6       Modified
   Bit  5       Field Selection
   Bit  4       Repeat
   Bit  3       Blinking
   Bit  2       Reverse Video
   Bit  1       Right Justification
   Bit  0       Reserved

Byte 1
   Bit  7       Reserved for color
   Bit  6       Reserved
   Bit  5       Protection
   Bit  4       Alphabetic-Only
   Bit  3       Numeric-Only
   Bits 0-2     Intensity
```

where:

If the Function-Key bit is set, the sender supports the FUNCTION-KEY and ENABLE-FUNCTION-KEY subcommands.

If the Modified bit is set, the sender supports the FORMAT-DATA subcommand’s Modified attribute and the

TRANSMIT-MODIFIED subcommand.

If the Field-Selection bit is set, the sender supports the FORMAT-DATA subcommand's Selectable attribute and the SELECTED-FIELD subcommand.

If the Repeat bit is set the sender supports the REPEAT subcommand.

If the Blinking bit is set, the sender requests or provides the ability to emphasize a string of characters by causing them to blink when displayed. (See the FORMAT-DATA subcommand.)

If the Reverse-Video bit is set, the sender requests or provides the ability to emphasize a string of characters by "reversing their video image". If characters are normally displayed as dark characters on a light background, they are reversed and displayed as light characters on a dark background, or vice versa. (See the FORMAT-DATA subcommand.)

If the Right-Justification bit is set, the sender requests or provides the ability to cause data entered in a field to be right justified within the field. (See the FORMAT-DATA subcommand.)

If the Protection bit is set, the sender requests or provides the ability to protect certain fields displayed on the DET screen from being altered by the user and supports the ERASE-UNPROTECTED, FIELD-SEPARATOR, and TRANSMIT-UNPROTECTED subcommands. (See the FORMAT-DATA subcommand.)

If the Alphabetic-Only bit is set, the sender requests or provides the ability to constrain the user of the DET such that only alphabetic data may be entered into certain fields. (See the FORMAT-DATA subcommand.)

If the Numeric-Only bit is set, the sender requests or provides the ability to constrain the user of the DET such that only numeric data may be entered into certain fields. (See the FORMAT-DATA subcommand.)

The Intensity parameter is three bits wide and is interpreted as a positive binary integer indicating the number of visible levels of intensity that the sender requests or provides for displaying data. (See the

FORMAT-DATA subcommand.)

Reserved bits represent format facilities that are not defined for DODIIS implementations; therefore, no descriptions are provided. Reserved bits must be zeroed to indicate non support of the associated format facilities.

EDIT SUBCOMMANDS. Edit subcommands are sent by the application to position the cursor on the DET screen.

IAC SB DET MOVE-CURSOR <x><y> IAC SE

subcommand code: 5

This subcommand positions the DET cursor at screen location (x,y). the <x> and <y> parameters are positive eight bit binary integers representing the character and line positions, respectively, of a DET screen location. Values of x range from zero (0) through M-1, where M is the DET screen width in characters. Values of y range from zero (0) through N-1, where N is the DET screen length in lines.

IAC SB DET HOME-CURSOR IAC SE

subcommand code: 12

This subcommand positions the cursor at DET screen address (0,0). It is equivalent to the MOVE-CURSOR subcommand, where x=0 and y=0.

TRANSMIT SUBCOMMANDS. Transmit subcommands are sent by the application to request data from the DET or by the terminal to identify data returned from the DET.

IAC SB DET READ-CURSOR IAC SE

subcommand code: 17

This subcommand requests return of the DET cursor position. Use of this subcommand requires facility negotiation; see the EDITFACILITIES subcommand, Read-Cursor bit.

IAC SB DET CURSOR-POSITION <x><y> IAC SE

subcommand code: 18

This subcommand returns cursor position in response to a

READCURSOR subcommand. The <x> and <y> parameters are eight bit binary integers representing the cursor's position. The <x> and <y> parameters are positive eight bit binary integers representing the character and line positions, respectively, of a DET screen location. Values of x range from zero (0) through M-1, where M is the DET screen width in characters. Values of y range from zero (0) through N-1, where N is the DET screen length in lines. Use of this subcommand requires facility negotiation; see the EDIT-FACILITIES subcommand, Read-Cursor bit.

IAC SB DET TRANSMIT-SCREEN IAC SE

subcommand code: 20

This subcommand requests return of all characters on the DET screen beginning at cursor position (0,0). M x N characters, where M is the DET screen width in characters and where N is the DET screen length in lines, are returned with a SPACE character returned for each character in the unwritten areas (the areas between defined fields). FIELD-SEPARATOR and DATA-TRANSMIT subcommands are not required to delimit or identify fields.

IAC SB DET TRANSMIT-UNPROTECTED IAC SE

subcommand code: 21

This subcommand requests return of all characters in unprotected fields. Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand, Protection bit.

IAC SB DET TRANSMIT-MODIFIED IAC SE

subcommand code: 27

This subcommand requests return of all characters in modified fields. Modified fields are fields that have the Modified attribute set (see FORMAT-DATA subcommand) as well as fields actually modified by the user. Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand, Modified bit.

IAC SB DET DATA-TRANSMIT <x><y> IAC SE

subcommand code: 28

This subcommand identifies a field returned in response to a TRANSMIT-MODIFIED subcommand. The <x> and <y> parameters are positive eight bit binary integers indicating the cursor position of the field that follows the DATA-TRANSMIT subcommand. This subcommand may precede the first field of a transmission with subsequent fields separated by the FIELD-SEPARATOR subcommand or it may precede each field. Use of this subcommand requires facility negotiation; see the TRANSMIT-FACILITIES subcommand, Data-Transmit bit.

ERASE SUBCOMMANDS. Erase subcommands are used by the application to erase the DET screen or selected DET screen areas. In performing erase operations, the erased characters are replaced with SPACE characters.

IAC SB DET ERASE-SCREEN IAC SE

subcommand code: 29

This subcommand erases all characters from the DET screen. All fields regardless of their attributes are deleted. The cursor position after the operation is at (0,0). If the protection attribute has been negotiated, the erased screen contains protected SPACE characters.

IAC SB DET ERASE-UNPROTECTED IAC SE

subcommand code: 35

This subcommand erases all characters in the unprotected fields of the DET screen. This subcommand replaces field contents with SPACE characters; field attributes and sizes are not changed. The cursor position after the operation is at the beginning of the first unprotected field or, if there is no unprotected field, at (0,0). Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand, Protection bit.

FORMAT SUBCOMMANDS. The format subcommands are used by the application to define the fields of a form and by the terminal to delimit fields sent from the DET.

IAC SB DET FORMAT-DATA <format map><count> IAC SE

subcommand code: 36

This subcommand defines the attributes and size of a DET field. The <format map> parameter defines the field attributes and the

<count> parameter defines the field size. The field starts at the position of the cursor when the subcommand is acted upon. The next <count> data characters in the data stream fill the field.

The <format map> parameter is two eight bit bytes and contains the following:

```
Byte 0
   Bit  7            Blinking
   Bit  6            Reverse Video
   Bit  5            Right Justification
   Bits 3-4          Protection
   Bits 0-2          Intensity

Byte 1
   Bits 5-7          Reserved
   Bits 2-4          Reserved for color
   Bit  1            Modified
   Bit  0            Selectable
```

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, characters entered into the field by the user should be right justified.

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

0 No protection. Any valid DET data character may be entered in the field.

1 Protected. No data may be entered in the field.

2 Alphabetic-only. Only the alphabetic characters (A-Z and a-z) or the space character may be entered in the field.

3 Numeric-only. Only the numeric characters (0-9),

the plus sign (+), the minus sign (-), the decimal point (.) or the space character may be entered in the field.

The Intensity attribute is three bits wide and indicates the brightness to be used when displaying the characters in or entered into the field <count> characters wide. The available number of visible intensity levels should have been negotiated using the FORMAT-FACILITY subcommand. A value of zero (0) indicates that brightness should be OFF; that is, characters in or entered into the field should not be displayed. The values 1-7 indicate relative brightness; the exact algorithm for mapping these values to the available levels of intensity is left to the implementors.

If the Modified bit is set, the field is considered to have been modified and will be returned, along with any user modified fields.

If the Selectable bit is set, the field is a candidate for field selection using the DET field selection device.

The <count> parameter is two bytes and should be interpreted as a positive 16-bit binary integer that defines the field size. The high order bit is transmitted first. Data, not in the scope of the count of a FORMAT-DATA subcommand, should be displayed with the default field attributes (no blinking, no reverse video, no justification, no protection, not modified, not selectable, and a visible intensity). Minimum field size is one (1) character. Maximum field size is determined by a field's starting location and the end of the screen or the start of the next field.

Use of field attributes requires facility negotiation; see the FORMAT-FACILITIES subcommand.

IAC SB DET REPEAT <count><char> IAC SE

subcommand code: 37

This subcommand permits compression of DET data by encoding strings of identical characters as the character and a repeat count. The <count> parameter is a positive 8-bit binary integer. The <char> parameter is a valid DET data character. Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand, Repeat bit.

IAC SB DET FIELD-SEPARATOR IAC SE

subcommand code: 39

This subcommand separates fields returned by the DET in response to TRANSMIT-MODIFIED or TRANSMIT-UNPROTECTED subcommands. Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand, Protection bit.

MISCELLANEOUS SUBCOMMANDS

IAC SB DET FUNCTION-KEY <code> IAC SE

subcommand code: 40

This subcommand transmits a user entered function key code. The <code> parameter is one byte that identifies the virtual function key entered. Function key <code> values range from 0 to 255. This subcommand is used in conjunction with the ENABLE-FUNCTION-KEY subcommand. Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand, Function-Key bit.

IAC SB DET ERROR <cmd><error code> IAC SE

subcommand code: 41

This subcommand allows a DET option implementation to report errors it detects to the corresponding TELNET process. The <cmd> parameter is one byte containing the subcommand code of the subcommand causing the error. The <error code> parameter is one byte containing a DET error code. (See Appendix 2 for DET error codes.)

Errors should be reported when detected. However, the implementation should attempt to carry out the intent of the subcommand or data in error.

IAC SB DET START-OUT-OF-CONTEXT-DATA IAC SE

subcommand code: 42

This subcommand precedes out-of-context data. The data following this subcommand and prior to the END-OUT-OF-CONTEXT-DATA subcommand is NOT part of the current form. The out-out-of-context data should be interpreted as NVT mode data (i.e., it may contain carriage return and line

feed characters) and should be displayed in a timely and non-destructive fashion.

IAC SB DET END-OUT-OF-CONTEXT-DATA IAC SE

subcommand code: 43

This subcommand indicates the end of the out-of-context data.

IAC SB DET ENABLE-FUNCTION-KEYS <key-map>IAC SE

subcommand code: 44

This subcommand enables (or disables) virtual function keys and indicates the application's data requirements on function key selection. The <key-map> parameter is a variable length byte string. Each byte contains four bit-pairs and each bit-pair represents a single function key. The first byte represents function keys zero (0) through three (3); the second byte, function keys four (4) through seven (7); and so on. Bit-pair values and there meanings are as follows:

0 The virtual function key is disabled (i.e., locked).

1 The virtual function key is enabled. Only the FUNCTION-KEY subcommand is returned on function key selection.

2 The virtual function key is enabled. All requested screen data and/or cursor position, as well as, the FUNCTION-KEY subcommand are returned on function key selection.

3 Undefined.

Function keys not explicitly represented in the bitmap are disabled (i.e., they are assumed to have a bit-pair value of zero (0)).

Use of this subcommand requires facility negotiation; see the FORMAT-FACILITIES subcommand; Function-Key bit.

IAC SB DET SELECTED-FIELD <x><y> IAC SE

subcommand code: 45

This subcommand identifies a user selected field. The <x> and <y> parameters are the cursor position of the character selected from within a selectable field (see the FORMAT-DATA

subcommand, Selectable attribute.) Use of this subcommand requires negotiation; see the FORMAT-FACILITIES subcommand, Field-Selection bit.

3. Default and Minimal Implementation

Default.

WONT DET -- DONT DET

If the DET option cannot be negotiated, the connection is not operated in DET mode.

Minimal DET Implementation.

The minimal DET implementation consists of all DET subcommands that may be used without prior negotiation. These subcommands are as follows:

EDIT-FACILITIES
ERASE-FACILITIES
TRANSMIT-FACILITIES
FORMAT-FACILITIES
MOVE-CURSOR
HOME-CURSOR
ERASE-SCREEN
TRANSMIT-SCREEN
FORMAT-DATA
ERROR
START-OUT-OF-CONTEXT-DATA
END-OUT-OF-CONTEXT-DATA

DODIIS DET implementation requirements.

The minimal DET implementation set of subcommands is not broad enough to support forms interactions between DODIIS terminals and DODIIS applications. Therefore, DODIIS implementations of the DET option must support additional DET subcommands.

DODIIS terminal (User Host) implementations must implement and support all of the DET subcommands contained in Section 2, as well as those DET attributes supported by the terminal hardware and any DET attributes easily emulated in software. DODIIS application (Server Host) implementations must implement and support those DET subcommands and attributes required by its applications.

DODIIS implementation recommendations are contained in the table that follows. DODIIS implementors are cautioned that failure to provide recommended support may limit interoperability.

Recommended DET support levels for DODIIS implementations

DET SUBCOMMANDS	USER HOST SUPPORT LEVEL	SERVER HOST SUPPORT LEVEL
EDIT-FACILITIES	send & receive	send & receive
ERASE-FACILITIES	send & receive	send & receive
TRANSMIT-FACILITIES	send & receive	send & receive
FORMAT-FACILITIES	send & receive	send & receive
REPEAT	send & receive	send & receive
ERROR	send & receive	send & receive
MOVE-CURSOR	receive only	send only
HOME-CURSOR	receive only	send only
READ-CURSOR	receive only	send only
TRANSMIT-SCREEN	receive only	send only
TRANSMIT-UNPROTECTED	receive only	send only
TRANSMIT-MODIFIED	receive only	send only
ERASE-SCREEN	receive only	send only
ERASE-UNPROTECTED	receive only	send only
FORMAT-DATA	receive only	send only
START-OUT-OF-CONTEXT-DATA	receive only	send only
END-OUT-OF-CONTEXT-DATA	receive only	send only
ENABLE-FUNCTION-KEYS	receive only	send only
CURSOR-POSITION	send only	receive only
DATA-TRANSMIT	send only	receive only
FIELD-SEPARATOR	send only	receive only
FUNCTION-KEY	send only	receive only
SELECTED-FIELD	send only	receive only
DET ATTRIBUTES		
Blinking	(1)	(2)
Reverse video	(1)	(2)
Right justification	(1)	(2)
Protection	required	(2)
Alphabetic-only protection	(1)	(2)
Numeric-only protection	(1)	(2)
Intensity level > 1	(1)	(2)
OTHER		
Page size (lines)	24-48	
Line size (characters)	80	

Function keys (number) 64

(1) Implement if supported by terminal hardware.
(2) Implement if required by the application.

4. Motivation for the option

In 1981, the TELNET DET option (RFC 732) was selected as the protocol to support interactions between DODIIS forms applications and DODIIS forms terminals. The intent was to foster a high degree of interoperability between DODIIS hosts with forms applications and terminals. Since that time, the DET option has been and is being implemented by several independent organizations within the DODIIS community.

Motivated by concern that the independently developed implementations of the DET option may not interoperate with one another, DODIIS implementors met to identify DODIIS implementation requirements and to resolve implementation issues that affect interoperability.

This document attempts to present the agreements and recommendations of the DODIIS implementors.

5. Description and Implementation Rules

The DODIIS DET model.

The conceptual model of the DODIIS DET is that of a half-duplex, forms oriented device with the following:

a. A rectangular screen for displaying protected and unprotected data (a form) and optional capability to support blinking, reverse video, and up to seven display intensity levels.

b. A keyboard and onboard mechanisms for editing unprotected fields of a form and returning the modified fields.

c. Function keys that may be enabled and disabled on a key-by-key basis by the application.

d. A field selection device, similar to a light pen, that permits user selection of characters within appropriately identified "selectable" fields.

The DODIIS DET screen has default sizes of 80 characters and 24 lines. These defaults may be changed through negotiation using the Output Line Width and the Output Page Size options. When the parties

cannot agree on screen size through negotiation, the default values will be used. By agreement, DODIIS terminal (User Host) implementations of DET will support page sizes of 24 to 48 lines.

The next writing position (x,y) on the DET screen is indicated by a special display character called the cursor, where x is the position of a character on a line and y is the line position on the DET screen. Values of x range from 0 (the left most character position on the line) to M-1, where M is the line length. Values of y range from 0 (the top most line on the screen) to N-1, where N is the page length. The cursor may be moved to any position on the DET screen without disturbing the characters already displayed.

Valid field data for DET forms are the displayable ASCII character codes in the range 32 through 126 decimal and character 7 "BELL".

Negotiating the DET option

The DET option is negotiated when either party REQUESTS use of the DET option and the other party AGREES to its use. The DET option is requested by sending a DO DET and WILL DET and is accepted by sending a WILL DET and DO DET. (In the spirit of TELNET negotiation, the DET option must be negotiated for both directions on the connection.)

Several TELNET options conflict with the DET option. Therefore, when the DET option is negotiated, the following TELNET options should be refused (or explicitly terminated): Echo, Suppress Go-Ahead, and Binary. (The Suppress Go-Ahead is the default state of DODIIS TELNET connections when they are first established.)

DET facilities negotiation

All implementations of the DET option are required to support the minimal DET implementation described in Section 3. In addition, DODIIS implementations are required to support subcommands and attributes that are consistent with DODIIS implementation requirements. Before any of these additional DET facilities may be used, an implementation must negotiate with its correspondent for permission to use them.

The four facility subcommands (EDIT-FACILITIES, ERASE-FACILITIES, TRANSMIT-FACILITIES, and FORMAT-FACILITIES) are used to negotiate DET subcommands and attributes. This negotiation consists of an exchange of facility subcommands and may be viewed as the terminal (User Host) indicating the facilities it provides and the application program (Server Host) indicating the facilities it desires. The facilities that are jointly supported (and may be

used) are arrived at by forming the logical intersection of the facility map that was sent with the facility map that was received. (For the intensity attribute, the lesser of the number of intensity levels sent and the number of intensity levels received will be used.) An implementation must record the currently agreed upon set of subcommands and attributes. Only subcommands and attributes reflected in that set may be used without further exchange of facility subcommands.

Either party or both parties may initiate facilities negotiation without confusion as long as care is taken to avoid non-terminating negotiation loops. In particular, if you initiate negotiation by sending a facility subcommand, you must remember that you did initiate the negotiation. On receipt of a facility subcommand; if you initiated the negotiation, no response is required and the negotiation is complete; if you did not initiate the negotiation, you must respond by sending the appropriate facility subcommand to the requester. (Note that there is no requirement to negotiate facilities one class at a time and that the awareness of who initiated the negotiation must be maintained for each of the facility subcommands.)

A TELNET implementation responding to a facility subcommand is not required to compute the logical intersection of the maps before responding. It should respond as quickly as possible with a facility map indicating all facilities of that class that it supports. There is no confusion since both parties compute the set of supported subcommands and attributes in the same fashion. Note that while both parties must agree to the use of the optional subcommands and attributes, either party may disable use at any time by merely sending the appropriate facility subcommand. Further, there are no restrictions on when facilities may be sent.

CAUTION:

All facilities maps contain reserved bits. These reserved bits must be zeroed when facility maps are sent to indicate non support and/or ignorance of the associated facility. The reserved bits may be defined in the future.

General DET Interaction

In the general interaction, the application implementation constructs a form, negotiates the desired options, indicates the required responses, and sends the TELNET GO-AHEAD. The GO-AHEAD signals that the form construction is complete and that the DET

keyboard may be unlocked to permit a user response.

The user normally responds by editing the unprotected areas of the form and signaling "form-complete", entering a function key, electing a field, or performing a combination of the preceding. In each case, the terminal implementation sends the DET subcommands indicating the user's response and returns the GO-AHEAD. The GO-AHEAD signals the end of the user response.

The form, as edited by the user, remains on the virtual screen so the application may continue the interaction by altering the form.

Form construction

The application implementation constructs a form on an erased screen by defining each of the fields in the form. The DET fields are defined by their starting cursor position, size, attributes, and contents (data).

A field's starting cursor position is the cursor position of the first character in the field. The cursor may be positioned explicitly by the MOVE-CURSOR subcommand or it may be positioned implicitly by field data or other DET subcommands (e.g., ERASESCREEN and ERASE-UNPROTECTED).

Field size, attributes, and contents may be defined using the FORMAT-DATA subcommand followed by field data. Alternatively, a field with default attributes may be defined using only the field data. In this case, field size is the data string length. The data string is terminated by the GO-AHEAD or any DET subcommand, except the REPEAT subcommand.

There are no restrictions on attribute combinations that might be applied to a field even though some combinations may not be supported by terminal hardware. The terminal implementation should display the field with a "reasonable" combination of attributes. There is an error code that might be returned when an "unsupported combination of format attributes" is detected. It is not clear what the application should do about the error. In any event, this condition should not provoke session termination.

Field contents (data) are restricted to printable ASCII characters and "BELL" (codes 32 through 126 and 7 decimal). It is the responsibility of the application implementation to properly translate carriage returns, line feeds, tabs, etc. to the appropriate DET subcommands.

The maximum number of fields a screen might contain is the screen

size in characters (the product of characters per line and lines per screen).

Fields may not overlap. That is, a new field may not start or end within a previously defined field. However, overwriting of a field to change its attributes or contents is permitted.

There are no restrictions on the order in which a form is built (e.g., left-to-right and top-to-bottom); the terminal implementation must be prepared to handle any order. Terminal implementations are encouraged to display data as it arrives to accommodate applications that persist in displaying status updates on the task(s) they are performing.

If an application elects to modify a user edited form, it must properly position the cursor making no assumptions about where the user might have left the cursor. Further it must exactly overwrite the existing fields.

When form construction is complete, the application indicates its response requirements by sending the appropriate transmit subcommand. It may send TRANSMIT-SCREEN, TRANSMIT-UNPROTECTED, or TRANSMIT-MODIFIED to request data and/or it may send READ-CURSOR to request cursor position. TRANSMIT-MODIFIED should be used whenever possible to minimize the volume of data transmitted between user and server hosts.

Form response

A form response is generated by the terminal implementation when the user signals "form-complete" or enters an enabled function key. The data returned are determined by the application through the transmit subcommands. If no transmit subcommand was sent the Modified and Protection attributes are used to determine an implied transmit subcommand. If the Modified attribute has been negotiated, assume TRANSMIT-MODIFIED. If the Protection attribute has been negotiated but the Modified has not, assume TRANSMITUNPROTECTED. If neither has been negotiated, assume TRANSMITSCREEN. (The intent is to achieve transmission efficiency by returning the smallest amount of data permitted by the in-force DET attributes.)

CAUTION:

With TRANSMIT-MODIFIED the terminal implementation must return all fields marked with the Modified attribute in addition to fields actually modified by the terminal user.

Returned fields are identified and delimited using the DATATRANSMIT and/or FIELD-SEPARATOR subcommands. The DATA-TRANSMIT subcommand indicates the cursor address of the field that follows it and there are no restrictions on the order in which fields are returned. The FIELD-SEPARATOR subcommand conveys left-to-right and top-to-bottom field ordering. Data not preceded by one of these subcommands is assumed to be the first unprotected field in the form. A FIELD-SEPARATOR followed by FIELD-SEPARATOR indicates a field was unchanged and not returned.

Unless otherwise restricted by Numeric-only or Alphabetic-only attributes, data entered into unprotected fields is restricted to the printable ASCII characters and "BELL" (codes 32 through 126 and 7 decimal); no other characters are permitted.

Function keys

By general agreement, DODIIS terminal implementations will support 64 function keys (key values 0 through 63). Information on mapping function keys to application functions is the responsibility of the application and should be provided to the terminal user in the form of user documentation.

The application enables and disables the function keys and indicates its form response requirements by sending the ENABLEFUNCTION-KEY subcommand. The terminal implementation validates function key selections based on information received in the ENABLE-FUNCTION-KEY bitmap. When an enabled function key is entered, the terminal returns a form response (if indicated in the bitmap), a FUNCTION-KEY subcommand, and the GO-AHEAD.

Virtual function keys are part of the DET’s virtual keyboard and are "locked" when the application has the GO-AHEAD. Since the terminal sends the GO-AHEAD when a function key is entered, entering a function key "re-locks" all function keys until the GO-AHEAD is returned.

Field selection

Any character within a field having the Selectable attribute is a candidate for selection. When selection is made, the terminal returns a SELECTED-FIELD subcommand identifying the character position selected. Multiple selections are permitted; however, the ordering of the selections need not be preserved. Field selection does not cause the GO-AHEAD to be sent. The GO-AHEAD must be sent as a result of another user action such as a function key entry or "form-complete" indication. Field selection is disabled when the application has the GO-AHEAD.

Out-of-context data

The out-of-context-data subcommands identify data that is clearly not in the context of the form interaction. It is a convenient not in the mechanism for sending ARE-YOU-THERE responses or host advisory messages to the user without disturbing the DET's virtual screen or altering the context of the form interaction.

The application may send out-of-context data at anytime. The data must be preceded by the START-OUT-OF-CONTEXT-DATA subcommand and followed immediately by the END-OUT-OF-CONTEXT-DATA subcommand. The out-of-context data should contain carriage returns and line feeds to facilitate formatting. The sender should limit the amount of data sent, since most terminal implementations must buffer the data prior to displaying it. The terminal implementation should display the data to the user in a timely fashion. The data is for display only, no user response is required, and there is no mechanism for user response.

Line Discipline

The subject of DET and line discipline (controlling the connection using the GO-AHEAD) causes a bit of confusion. The following rules apply to GO-AHEAD and the DET option:

When DET is negotiated, the application assumes the GO-AHEAD. GO-AHEAD is never passed implicitly; it is always passed explicitly.

When the application has the GO-AHEAD, the terminal implementation may send TELNET commands (INTERRUPT-PROCESS, ABORT-OUTPUT, BREAK, and ARE-YOU-THERE). Nothing else is valid.

When the terminal has the GO-AHEAD, the application may send out-of-context data or MOVE-CURSOR and FORMAT-DATA subcommands to update protected fields. Nothing else is valid. (The terminal implementation must display the out-of-context data and the field updates as soon as convenient.)

The terminal implementation sends the GO-AHEAD, without further action on the part of the terminal user, when an enabled function key or a "form-complete" is entered.

Since the terminal user must take explicit action to return the GO-AHEAD to the application, instances will occur when the user has the GO-AHEAD but the application needs it to display a new form. (This is most likely to occur when the user enters an

INTERRUPT PROCESS.) When it does occur, the application should send an out-of-context-context message requesting the user to enter a "form-complete". If the user cooperates, the application can ignore any associated form response and regain control of the connection to display its form.

The line discipline described here is more rigorous than that described for NVT in MIL-STD-1782. These rules apply only when operating in DET mode. At other times, the descriptions contained in MIL-STD-1782 apply. This distinction is necessary to ensure interoperability with non-DET implementations of TELNET.

Standard TELNET control functions

The TELNET control functions, ERASE CHARACTER and ERASE LINE, are NOT required and should not be sent in DET mode.

Other implementation notes

a. The DODIIS DET conceptual model does not support character editors or basic scrolling applications.

b. Implementors are cautioned that DET subcommand parameters (e.g., facilities maps) may take on the value of the IAC character and must be replicated if they are to be properly interpreted.

c. Principle of Robustness: "Be conservative in what you send; be liberal in what you accept from others."

APPENDIX 1 - DET OPCODES AND SUBCOMMAND SYNTAX.

OPCODE	SUBCOMMAND SYNTAX
1	EDIT-FACILITIES <facility 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>
12	HOME-CURSOR
17	READ-CURSOR
18	CURSOR-POSITION <x><y>
20	TRANSMIT-SCREEN
21	TRANSMIT-UNPROTECTED
27	TRANSMIT-MODIFIED
28	DATA-TRANSMIT <x><y>
29	ERASE-SCREEN
35	ERASE-UNPROTECTED
36	FORMAT-DATA <format map><count>
37	REPEAT <count><character>
39	FIELD-SEPARATOR
40	FUNCTION-KEY <code>
41	ERROR <cmd><error code>
42	START-OUT-OF-CONTEXT-DATA
43	END-OUT-OF-CONTEXT-DATA
44	ENABLE-FUNCTION-KEYS <key-map>
45	SELECTED-FIELD <x><y>

APPENDIX 2 - DET ERROR CODES

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