DECnet
Digital Network Architecture (Phase IV)

Network Virtual Terminal Foundation Services

Order No. AA-DY89A-TK

digital

software

DECnet
Digital Network Architecture (Phase IV)

Network Virtual Terminal Foundation Services

Order No. AA-DY89A-TK

December 1984

This document describes the Foundation (FOUND) services, which provide the mode-independent terminal handling services required by the Terminal Software Architecture (TSA). FOUND is a sublayer within the Application layer of the Terminal Software Architecture (TSA) — and TSA is part of the Digital Network Architecture.

SUPERSESSION/UPDATE INFORMATION: This is a new manual.

VERSION: 2.0

digital equipment corporation • maynard, massachusetts

AA-DY89A-TK

First Printing, December 1984

CONTENTS

1.0 INTRODUCTION

This document describes a model for distributed terminal-handling services in Digital systems. These services are independent of the specific use (e.g., command handling, forms, editing) being made of the terminal and include connection management, mode changing, and local characteristics management.

Foundation services are part of a larger model for terminal handling within Digital systems. This model is defined by the Digital Terminal Software Architecture.

It is the intent of this specification to define the following:

1. The services (viz., interface functions or primitive operations) and semantics (but not the syntax) of the services provided by this model for foundation services within Digital's Terminal Software Architecture

2. The communication protocol (both syntax and semantics) used by this model to provide the defined services

It is stongly suggested that an implementation of this specification create a sharable interface corresponding to the interface to logical terminal services as described herein.

1.1 Requirements, Goals, and Non-goals

The requirements of the foundation services are:

- o Compatibility -- to be compatible with the DECnet Network Virtual Terminal specification, Version 1.0, dated 13 April 1979

- o Connection Management -- to allow a logical terminal in a server system to connect to a specified host; to allow a host system to connect to a specified logical terminal

- o Mode Changing -- to allow a pair of mode modules to transfer control of a connection to a second pair of mode modules

- o Terminal Management -- to allow a module implementing terminal management functions to intercept input from a physical terminal

- o Extensibility -- to allow evolution of this specification, particularly the protocol portion.

The goal of the foundation services is:

- o Simplicity -- to be easy to understand and implement.

The non-goal of the foundation service is:

- o Expansion -- to contain functions other than those required to implement the first version of the Digital command terminal.

1.2 **Relation to Digital's Terminal Software Architecture**

This document is the specification of the foundation layer of Digital's Terminal Software Architecture.

1.3 **Relation to Digital's Network Architecture**

This specification describes services considered to be in the application layer of Digital's Network Architecture.

1.4 **Definition of Character Set**

This specification is intended for use with Digital's 8-bit coded character set. The term "character" means an 8-bit value from this character set. This character set defines the names of characters referred to in this specification (e.g., BEL) and it imposes certain requirements on "system ports", which are points at which character set conversion takes place. Such system ports are believed to be within the scope of foundation services. In particular, implementations of foundation services should include fallback presentation and conversion between 7-bit and 8-bit environments, as described in the standard.

2.0 **MODELS**

There are two components to the terminal software architecture foundation services: logical terminal services and terminal communication services.

Figure 2-1 presents a model of the complete terminal software architecture. This model shows the distribution of functions between a host system and a server system. The modules labeled "logical terminal services" and "terminal communication services" constitute the foundation services of the architecture. (The remaining modules are shown in the figure as an example of the modules that would use the foundation services.)

```
         HOST SYSTEM                  SERVER SYSTEM
+--------------------------+   +--------------------------------+
|                          |   |                                |
|  +-------------------+   |   |                                |
|  |   APPLICATION     |   |   |                                |
|  +-------------------+   |   |                                |
|      |        |          |   |                                |
|      |        V          |   |                                |
|      |  +-------+        |   |   +-------+                    |
|      |  | FORMS |  . . . |   |   | FORMS | . . .              |
|      |  | MODE  |        |   |   | MODE  |                    |
|      |  +-------+        |   |   +-------+                    |
|      |        |          |   |     |   |                      |
|      V        |          |   |     |   |                      |
| +----------+  |          |   |     | +----------+             |
| |  COMMON  |  |          |   |     | |  COMMON  |             |
| | TERMINAL |  |          |   |     | | TERMINAL |             |
| | SERVICES |  |          |   |     | | SERVICES |             |
| +----------+  |          |   |     | +----------+             |
|      |        |          |   |     |     |   |                |
|      |        |          |   |     |     |   +-------+        |
|      |        |          |   |     |     |           |        |
|      V        V          |   |     V     V           V        |
| +----------------------+ |   | +-------------+ +-----------+  |
| |       TERMINAL       | |   | |  TERMINAL   | |  LOGICAL  |  |
| |     COMMUNICATION    | |   | |COMMUNICATION| | TERMINAL  |  |
| |       SERVICES       | |   | |  SERVICES   | | SERVICES  |  |
| +----------------------+ |   | +-------------+ +-----------+  |
|            |             |   |         |             |        |
|            V             |   |         V             |        |
| +----------------------+ |   |  +-------------+      |        |
| |        NETWORK       | |   |  |   NETWORK   |      |        |
| |     COMMUNICATION    | |   |  |COMMUNICATION|      |        |
| |       SERVICES       | |   |  |  SERVICES   |      |        |
| +----------------------+ |   |  +-------------+      |        |
|            |             |   |         |             |        |
+------------|-------------+   +---------|-------------|--------+
             |                           |             |
             |       +----------------+  |           physical
             +-------|    NETWORK     |--+          terminal(s)
                     +----------------+
```

Figure 2-1 Distributed System Architectural Model

2.1 Logical Terminal Services

The logical terminal services module has three interfaces: (1) one to modules in the mode access layer, (2) one to a terminal management module, and (3) one to a terminal user. These interfaces as well as pertinent internal data bases and queues are shown in Figure 2-2. All modules and stuctures shown in Figure 2-2 are in the server system.

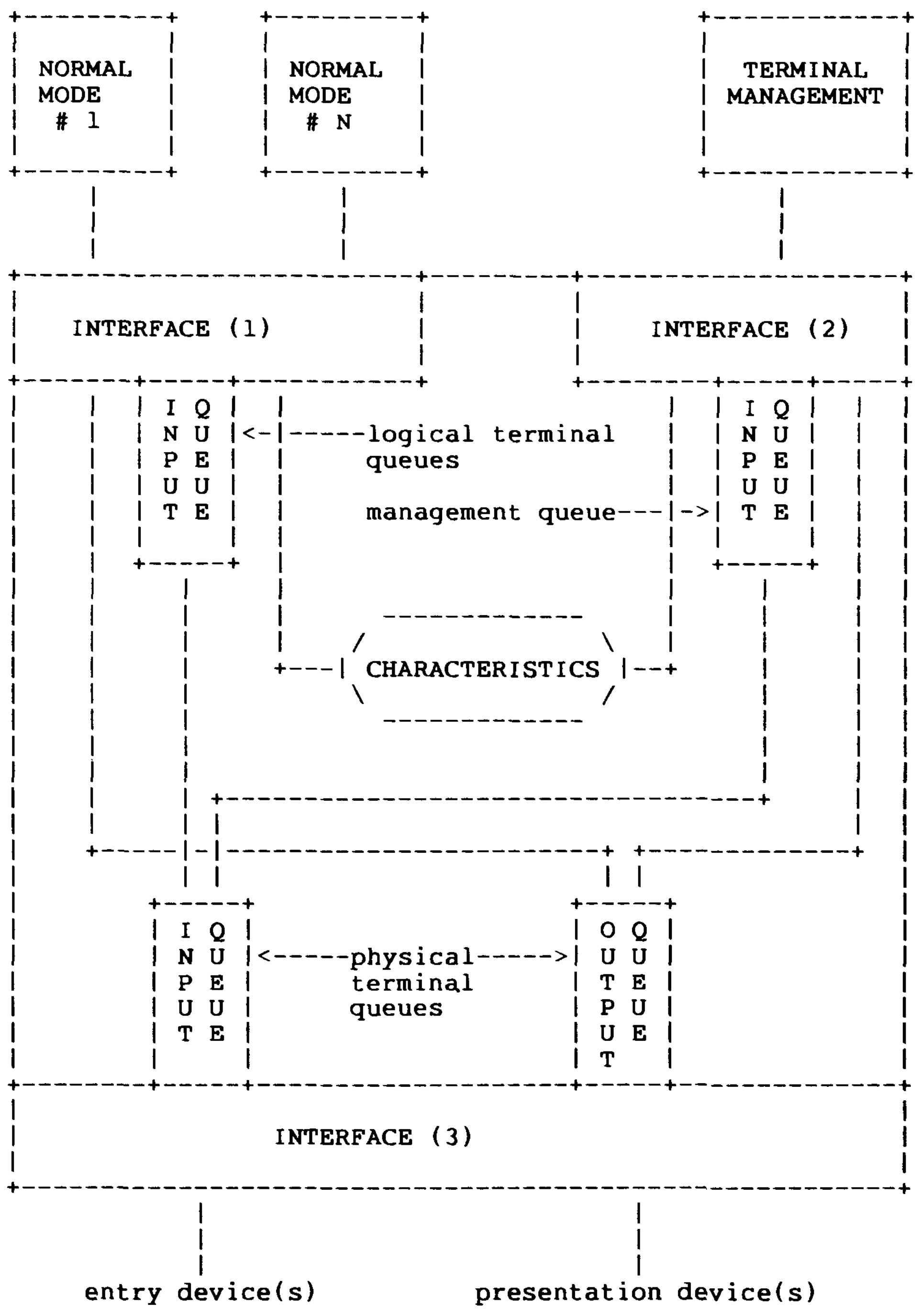

Figure 2-2 Logical Terminal Services Model

2.2 **Terminal Communication Services**

There are two terminal communication services modules: one in the host system and one in the server system. These modules provide a distributed communication service between mode access modules in a host system and mode access modules in a server system as shown in Figure 2-3. This distributed service has two interfaces: one to mode access modules in the host system and one to mode access modules in the server system.

```
        HOST SYSTEM                          SERVER SYSTEM
+----------------------------+       +----------------------------+
|                            |       |                            |
|                            |       |                            |
|  +---------+  +---------+  |       |  +---------+  +----------+ |
|  | MODE    |  | MODE    |  |       |  | MODE    |  |  MODE    | |
|  | MODULE  |  | MODULE  |  |       |  | MODULE  |  |  MODULE  | |
|  |   #1    |  |   #2    |  |       |  |   #1    |  |   #2     | |
|  +---------+  +---------+  |       |  +---------+  +----------+ |
|       |            |      |       |       |            |       |
|       |            |      |       |       |            |       |
|  +-----------------------|- - -  |------------------------+    |
|  |                       |       |                        |    |
|  |  DISTRIBUTED TERMINAL |       | COMMUNICATION SERVICES |    |
|  |                       |       |                        |    |
|  +-----------------------|- - -  |------------------------+    |
|                            |       |                            |
+----------------------------+       +----------------------------+
```

Figure 2-3 Distributed Terminal Communication Services Model

The terminal communication services modules provide three functions: (1) logical connection of a resource (called a "portal") in the host system with a logical terminal (the logical connection is called a "binding"), (2) establishing and changing the mode of terminal operation, and (3) data transfer between the host and server systems.

3.0 **INTERFACES**

This section describes four interfaces: (1) the logical terminal services interface provided to higher level modules, (2) the terminal communication services interface provided to higher level modules, (3) the virtual circuit service interface used by the terminal communication services modules to communicate with each other in a network, and (4) the interface to a physical terminal from a server system.

Each interface function is described as being provided by a procedure. This interface definition technique is consistent with a software engineering model in which many processes execute in parallel, each requesting services from a collection of procedures. Such a model is not generally implementable as described but can be considered isomorphic to a model that may be implemented more directly. This technique is used in this specification rather than a technique that would be consistent with a different model (e.g., one in which several processes queue events to each other) for the following reasons.

- o This model explicitly states what can be considered to be an atomic operation and what cannot.
- o Race conditions tend to be more easily identified in this model.
- o Points in which two or more processes need to be synchronized are clearly identified. (They occur where procedures access data shared among several callers.)
- o Significant resource allocation failures are explicitly modeled.

Interface functions for each interface are shown throughout this section of the specification and are formatted as follows:

The function name is followed first by a list of arguments and return values that are enclosed in parentheses. The arguments appear first and are separated from the return values by a semicolon (;). Optional arguments are enclosed in brackets ([.....]).

Example:

READ-CHARACTER (logterm id; return,character)

The following list defines some of the return values that may appear with a function definition. Error return values whose definitions are obvious are not described -- these are: invalid portal id; invalid logical terminal id; invalid state, and invalid argument.

1. The term "buffer" syntactically refers to a combination of address and length information and, when data is being given to a procedure, semantically refers to the data contained in the memory defined by the address and length.

2. The term "logical terminal id" refers to a number that uniquely identifies a logical terminal in a server system.

3. The term "physical terminal id" refers to a number in the range (1 - maximum physical terminal id) that uniquely identifies a physical terminal in a server system.

4. The term "portal id" refers to a number in the range (1 - maximum portal id) that uniquely identifies a portal in a host system.

5. The term "character" refers to a value from a character set (see the terminal characteristics definitions in Section 3.1.1).

3.1 Logical Terminal Services

The primary functions of the logical terminal services module are to move characters between devices and higher-level modules, to store logical terminal characteristics, to switch a physical terminal's data stream between a terminal management module and a normal mode access module, and to act on certain terminal characteristics. These functions are described from the point of view of the users of the logical terminal services module.

3.1.1 Terminal Characteristics and Counters

3.1.1 Terminal Characteristics and Counters - Table 3-1 defines physical terminal characteristics managed by the logical terminal services module. A future version of this specification may define additional characteristics or enhance the definitions given below.

Table 3-1 Physical Terminal Characteristics

Characteristic	Meaning
Input-speed	The bit-per-second input rate, ignoring fractions, of the physical terminal. A 16-bit integer (the value 65,535 means any value greater than or equal to 65,535).
Output-speed	The bit-per-second output rate of the physical terminal. A 16-bit integer (the value 65,535 means any value greater than or equal to 65,535).
Physical-character-size	The bit-width of a character (a value of 5,6,7, or 8). See the comment following this table regarding the relation of this characteristic to the definition of character set.

(continued on next page)

Table 3-1 (Cont.): Physical Terminal Characteristics

Characteristic	Meaning
Eight-bit	Whether or not the eighth bit of 8-bit-wide characters is cleared. A Boolean value. TRUE = clear 8th bit (default value) FALSE = do not clear 8th bit NOTE For "pass-all" operation, the 8th bit is never cleared.
Parity-enable	Whether or not a parity bit is generated on output and checked on input. (This bit is in addition to the data bits and not counted in the Character-size characteristic.) When enabled, a parity bit is treated as the most significant bit of a character. A Boolean value. See the comment following this table regarding the relation of this characteristic to the definition of character set. TRUE = parity is enabled FALSE = parity is not enabled
Parity-type	The type of parity. One of the following: Value — Definition 1 — EVEN Parity 2 — ODD Parity 3 — CLEAR (forces parity bit to 0) 4 — SET (forces parity bit to 1)
Modem-present	Whether or not modem control signals (as reflected in the physical terminal interface) are present for this terminal. A Boolean value. TRUE = modem signals are present FALSE = modem signals are not present

(continued on next page)

Table 3-1 (Cont.): Physical Terminal Characteristics

Characteristic	Meaning
Auto-baud-detect	Whether or not the automatic baud detect algorithm is executed for this terminal. A Boolean value. TRUE = the automatic baud detection algorithm is executed FALSE = the automatic baud detection algorithm is not executed
Management-guaranteed	Whether or not a mode access module can disable the human physical terminal user's ability to enter terminal management mode (as described below). A Boolean value. TRUE = the user is guaranteed to be able to enter terminal management mode from the physical terminal FALSE = a mode access module may disable the entry of terminal management mode from the physical terminal
Terminal-management-enabled	Whether or not the host will allow the terminal user to exercise terminal management capabilities. A Boolean value. TRUE = terminal management is enabled (default) FALSE = terminal management is disabled NOTE This characteristic is forced TRUE if "management-guaranteed" is TRUE.
Switch-character-1	The first of the two characters that, on entry, cause a switch between normal and terminal management modes (see Section 3.1.5.2).
Switch-character-2	The second of the two characters that, on entry, cause a switch between normal and terminal management modes (see Section 3.1.5.2).

The combination of the Character-size and Parity-enable characteristics allow for the definition of 5-,6-,7-,8-, or 9-bit characters. However, the character set used with this specification contains only 8-bit characters (as defined in Table 3-1). On both output and input, a character with fewer than 8 bits is padded with high order zero bits to form an 8-bit character, and a 9-bit character has the high order bit truncated.

Table 3-2 defines physical terminal counters managed by the logical terminal services module.

Table 3-2 Physical Terminal Counters

Counter	**Meaning**
Connections	The number of times the physical terminal has entered the CONNECTED state. A 16-bit value.
Chars-sent	The number of characters sent to the physical terminal. A 32-bit value.
Chars-received	The number of characters received from the physical terminal. A 32-bit value.
Parity-errors	The number of parity errors. A 16-bit value.
Overruns	The number of overruns. A 16-bit value.
Framing-errors	The number of framing errors. A 16-bit value.
Seconds	The number of seconds since the counters were last zeroed. A 16-bit value.

Table 3-3 defines logical terminal characteristics managed by the logical terminal services module.

Table 3-3 Logical Terminal Characteristics

Characteristic	**Meaning**
Mode-writing-allowed	Whether or not the physical terminal characteristics defined in Table 3-1 or the logical terminal characteristics defined in this table (but not this characteristic) can be written by a mode access module. A Boolean value. TRUE = the characteristics can be written FALSE = the characteristics cannot be written

(continued on next page)

Table 3-3 (Cont.): Logical Terminal Characteristics

Characteristic	Meaning
Terminal-attributes	Attributes of the logical terminal. A bit map of length two bytes, with the bits defined as follows:

Bit	Definition
0	1 = known, 0 = unknown
1	1 = video, 0 = hardcopy
2-15	reserved

Characteristic	Meaning
Terminal-type	The "model" of the logical terminal. A string containing between 0 and 10 characters. This may or may not be the kind of physical terminal that corresponds to the logical terminal. It means that terminal-specific escape sequences or other traits are obtained when reading from or writing to the terminal from a normal mode access module. The strings for the terminal-type are to be obtained from the OPTION/MODULE LIST by taking the characters of the Digital Terminal model number up to but not including the first hyphen. The following strings are examples of "known" terminal types:

LA30
LA34
LA36
LA36RO
LA37
LA38
LA120
LA180
LS120
LT33
LT35
LT37
VK100
VT05
VT50
VT52
VT55
VT61
VT100
VT103
VT130

(continued on next page)

Table 3-3 (Cont.): Logical Terminal Characteristics

Characteristic	Meaning

Output-flow-control	Whether or not one type of flow control is exerted by the logical terminal services module on output. A Boolean value. (This value is independent of the Output-page-stop value described below.)
	
	TRUE = an input XOFF stops output; an input XON starts output
	FALSE = no interpretation of XOFF or XON on input

Output-page-stop	Whether or not a second type of flow control is exerted by the logical terminal services module on output. A Boolean value. (This value is independent of the Output-flow-control value described above.) TRUE = the end of a page stops output; an input XON starts output FALSE = no interpretation of end-of-page condition or XON input
Flow-character-pass-through	Whether or not an input character that affects output flow control (control-S or control-Q) is passed through or discarded on input. A Boolean value. TRUE = an enabled flow control character is passed to the mode access layer as a normal input character FALSE = an enabled flow control character is discarded on input
Input-flow-control	Whether or not flow control is exerted by the logical terminal services module on input. A Boolean value. TRUE = an XOFF is sent to the terminal when the logical terminal services module is becoming short of receive memory resources, and an XON is sent when the logical terminal module has more memory resources available

(continued on next page)

Table 3-3 (Cont.): Logical Terminal Characteristics

Characteristic	Meaning
Input-flow-control (cont.)	FALSE = no flow control is exerted on input
Loss-notification	Whether or not notification is sent to the physical terminal when an input character is lost due to lack of memory. A Boolean value. Even when this value is true, loss notification may not be possible either due to the server system's inablilty to detect all lost characters or due to a lack of memory resources for queueing the loss notification. TRUE = a BEL is sent to the terminal if an input character is lost FALSE = no notification is sent to the terminal if an input character is lost
Line-width	The width of a "line". "Line" has meaning in the logical terminal services module when the wrap characteristic indicates either hardware or software wrapping (see Wrap characteristic described below). A 16-bit integer.
Page-length	The length of a "page" as used by the "vertical tab and form feed expansion" algorithms (see the Vertical-tab and Form-feed characteristics). An 8-bit integer.
Stop-length	The length of a "page" as used by the "output page stop" algorithm (executed when the Output-page-stop characteristic is TRUE). An 8-bit integer.
CR-fill	The number of null characters automatically sent to the physical terminal after a carriage return (CR) is written. An 8-bit integer.
LF-fill	The number of null characters automatically sent to the physical terminal after a line feed (LF) is written. An 8-bit integer.

(continued on next page)

Table 3-3 (Cont.): Logical Terminal Characteristics

Characteristic	Meaning	
Wrap	An integer value indicating if character wrapping is provided on output and, if so, how it is provided. A future version of this specification may enhance this function. One of the following:	
	Value	**Definition**
	1	It is assumed that the hardware is not wrapping, the software is not wrapping, and no truncation of output is performed.
	2	It is assumed that the hardware is not wrapping, the software is not wrapping, but the software is discarding all data whose modeled horizontal position would be greater than Line-width.
	3	The hardware is wrapping, and the logical terminal services module is attempting to track horizontal position (this is identical to value 4 except that no carriage return or line feed characters are inserted in the output stream).
	4	The logical terminal services module is performing wrapping by inserting carriage return and line feed characters in the output stream.

(continued on next page)

Table 3-3 (Cont.): Logical Terminal Characteristics

Characteristic	Meaning
Horizontal-tab	An integer value indicating how a horizontal tab (HT) is handled on output. A future version of this specification may enhance this function. One of the following: Value — Definition 1 — The hardware is handling horizontal tabs, and the logical terminal services module is attempting to track horizontal position; it is assumed that horizontal tabs move the horizontal position to the next multiple of 8. 2 — The logical terminal services module is performing horizontal tab expansion by inserting spaces in the output stream to the next horizontal position that is a multiple of 8.
Vertical-tab	An integer value indicating how a vertical tab (VT) is handled on output. A future version of this specification may enhance this function. One of the following: Value — Definition 1 — The hardware is handling vertical tabs, and the logical terminal services module is attempting to track vertical position; it is assumed that vertical tabs move the vertical position to the next multiple of 11. 2 — The logical terminal services module is performing vertical tab expansion by inserting line feeds in the output stream to the next vertical position that is a multiple of 11.

(continued on next page)

Table 3-3 (Cont.): Logical Terminal Characteristics

Characteristic	Meaning	
Vertical-tab (cont.)	**Value**	**Definition**
	3	The logical terminal services module is converting a vertical tab to a form feed and handling it as specified by the Form-feed characteristic.
Form-feed	An integer value indicating how a form feed (FF) is handled on output. A future version of this specification may enhance this function. One of the following:	
	Value	**Definition**
	1	The hardware is handling form feeds, and the logical terminal services module is attempting to track vertical position.
	2	The logical terminal services module is performing form feed expansion by inserting line feeds in the output stream to a vertical position that is an integral multiple of Page-length lines from the last form feed.

3.1.2 **Normal Mode Access Interface** - The interface used by normal mode access modules contains .the following functions:

1. read the set of logical terminal id's
2. read a character
3. write a character
4. read a terminal characteristic
5. write a terminal characteristic
6. disable terminal management switchover
7. enable terminal management switchover
8. reset page-stop-position

The first two functions that follow allow a mode access module to obtain the active set of logical terminal id's. (These functions are an artifact of the subroutine modeling technique used in this specification. The information implied by them would probably be conveyed in a more straightforward manner in an implementation.)

GET-FIRST-LOGICAL-TERMINAL-ID (;logterm id)

logterm id	= The value of the "first" logical terminal id in the list of current logical terminal id's maintained by the logical terminal services module.

GET-NEXT-LOGICAL-TERMINAL-ID (;return, logterm id)

return	= One of the following: logical terminal id returned no more logical terminal id's
logterm id	= The value of the "next" logical terminal id in the list of current logical terminal id's maintained by the logical terminal services module.

READ-CHAR (logterm id; return, character)

return	= One of the following: success - character returned failure - no character returned failure - line break failure - framing error failure - parity error failure - receiver overrun

The character returned on success is removed from the logical terminal input queue. If the logical terminal services module detects an error when attempting to input from the underlying physical terminal, it queues the error on the logical terminal input queue (sequentially with respect to input data). The last four failure returns result from the removal of such an error indication from the input queue. If more that one error was detected simultaneously by the logical terminal services module, it only indicates a single error via this function. The error precedence is the same as in the failure list above (i.e., line break takes precedence over framing error, framing error takes precedence over parity error, and parity error takes precedence over receiver overrun). Line break, framing error, parity error, and receiver overrun all queue the character on which the error occurred as well as the error (this character may be required by some mode modules, e.g., CTERM).

WRITE-CHAR (logterm id, transparency, character; return, h-position, v-position)

transparency	= A Boolean value indicating whether or not the character should have an effect on horizontal and vertical position. One of the following: TRUE means HT, VT, and FF are not expanded, wrapping is not done, and the h-position and v-position values are returned as zero. FALSE means HT, VT, and FF are expanded, wrapping is done, and horizontal and vertical position are modeled.
return	= One of the following: success failure - insufficient resources
h-position	= The horizontal position after writing the character. A value in the range <0, Line-width - 1>.
v-position	= The vertical position after writing the character. A value in the range <0, Page-length - 1>.

A success return indicates that the character has been placed on the logical terminal output queue. Horizontal and vertical positions are discussed in Sections 3.1.5 through 3.1.5.3.

READ-CHARACTERISTIC (type,logterm id, selector; value)

type	= The type of characteristic to be read. One of the following: "physical" "logical"
logterm id	= A logical terminal id.
selector	= A value indicating which characteristic to read.
value	= The value of the selected characteristic.

WRITE-CHARACTERISTIC (type, logterm id, selector, value; return)

type	= The type of characteristic to be written. One of the following: "physical" "logical"

logterm id	= A logical terminal id.
selector	= A value indicating which characteristic to write.
value	= The value of the selected characteristic.
return	= One of the following:

success

failure - function disabled

failure - invalid value

This function may be used to write any physical terminal characteristic and any logical terminal characteristic except Mode-writing-allowed. This function fails for logical terminal characteristics if Mode-writing-allowed is false.

A terminal characteristic written via this function is used temporarily. Once the logical terminal binding has been broken, the terminal characteristics revert to those most recently established via the terminal management interface (discussed below).

DISABLE-MANAGEMENT-SWITCHOVER (logterm id; return)

return = One of the following:

success

failure - function disabled

This function disables the ability of the human terminal user to switch the associated physical terminal into terminal management mode. Its purpose is to allow transparent, binary input from a device (e.g., a cassette reader) connected to the server system via a physical terminal interface. The failure return indicates that the user has set the Management-guaranteed physical terminal characteristic TRUE.

ENABLE-MANAGEMENT-SWITCHOVER (logterm id)

This function reenables the terminal user to enter terminal management mode after a previous DISABLE-MANAGEMENT-SWITCHOVER function.

RESET-PAGE-STOP-POSITION (logterm id)

This function resets the "page-stop-position" variable (described in Section 3.1.5.3 on "Position Modeling") to zero. This function provides a facility whereby the mode module can control output stopping due to the page-stop-position variable reaching the Stop-length limit; in particular, it will allow the mode module to differentiate between output from the host and output due to local echoing of input. (For use with the CTERM mode module, it is intended that CTERM resets the page-stop-position variable on Reads after the prompt from the Start Read message has been echoed -- this will produce the same visual effect as seen on local terminals using page stop.)

NOTE

This function produces a race condition in the TSA Model as characters for output by the foundation service are queued to the foundation layer and there is no means of synchronizing the reset page-stop-position function and the prompt characters in the queue. However, this is a problem peculiar to the Model, and implementations should be able to solve it easily.

3.1.3 **Terminal Management Interface** - The interface used by the terminal management module contains the following functions:

1. read the maximum physical terminal id
2. read a character
3. write a character
4. read a terminal characteristic
5. write a terminal characteristic
6. define a logical terminal's name
7. free flow control
8. read the physical terminal connection state
9. hang up the physical terminal connection
10. disable the answering of a physical terminal connection
11. enable the answering of a physical terminal connection
12. read counter
13. clear counters

GET-MAX-PHYSICAL-TERMINAL-ID (; max id)

max id	= The maximum value of a physical terminal id.

READ-MANAGEMENT-CHAR (physical terminal id, character; return)

return = One of the following:

success - character returned
failure - no character returned
failure - line break
failure - framing error
failure - parity error
failure - receiver overrun

The character returned on success is removed from the management input queue. If the logical terminal services module detects an error when attempting to input from the underlying physical terminal, it queues the error on the management input queue (sequentially with respect to input data). The last four failure returns result from the removal of such an error indication from the input queue. If more than one error was detected simultaneously by the logical terminal services module, it only indicates a single error via this function. The error precedence is the same as in the failure list above (i.e., line break takes precedence over framing error, framing error takes precedence over parity error, and parity error takes precedence over receiver overrun). Line break, framing error, parity error, and receiver overrun all queue the character on which the error occurred as well as the error (this character may be required by some mode modules, e.g., CTERM).

WRITE-CHAR (physical terminal id, character; return)

return = One of the following:

success

failure - insufficient resources

A success return indicates that the character has been placed on the physical terminal output queue.

READ-CHARACTERISTIC (type, terminal id, selector; value)

type = The type of characteristic to be read. One of the following:

"physical"

"logical"

terminal id	= Either a physical terminal id or a logical terminal id, as selected by "type".
selector	= A value indicating which characteristic to read.
value	= The value of the selected characteristic.

WRITE-CHARACTERISTIC (type, terminal id, selector, value; return)

type	= The type of characteristic to be written. One of the following: "physical" "logical"
terminal id	= Either a physical terminal id or a logical terminal id, as selected by "type".
selector	= A value indicating which characteristic to write.
value	= The value of the selected characteristic.
return	= One of the following: success failure - invalid value

DEFINE-NAME (logterm id, name)

name	= A string of characters that defines the logical terminal's name, used in binding functions described in Section 3.2.3 (the maximum name length is implementation-specific but is in the range 6-40, inclusive).

FREE-FLOW-CONTROL (physical terminal id)

This function frees the output flow control state. It has the same effect as the entry of an XON from the physical terminal.

READ-PHYSICAL-CONNECTION-STATE (logterm id; state)

state	= The state of the connection to the physical terminal. One of the following: DON'T-ANSWER DISCONNECTED-Idle DISCONNECTED-Timeout CONNECTING CONNECTED-Active CONNECTED-Sync-Wait CONNECTED-CD-Wait CONNECTED-CTS-Wait

These states are further discussed in Section 3.1.3.1, Physical Connection States.

HANG-UP (physical terminal id)

This function causes the physical terminal connection to be disconnected (i.e., it "hangs up" the connection). The state becomes DISCONNECTED-Timeout. A new connection can be made after the timeout and the state goes to DISCONNECTED-Idle.

DISABLE (physical terminal id)

This function causes the physical terminal connection to be disconnected (i.e., it "hangs up" the connection) and inhibits the reconnection of the physical terminal. The state becomes DON'T-ANSWER.

ENABLE (physical terminal id)

This function allows a new physical terminal connection to be made. The state becomes DISCONNECTED-Idle if it was DON'T-ANSWER. This function is ignored if the state is not DON'T-ANSWER.

READ-COUNTER (physical terminal id, selector; counter)

selector	= The counter selector. Counters are described above with characteristics.
counter	= The returned counter value.

CLEAR-COUNTERS (physical terminal id)

This functions clears all counters associated with the physical terminal.

3.1.3.1 **Physical Connection States** - The "physical terminal" referred to above may really be hardware that is capable of communicating with a physical terminal and also capable of switching (i.e., "answering" and "hanging up"). The Modem-present characteristic is true for these physical terminals. The physical connection is modeled as having a state that can be read via the READ-PHYSICAL-CONNECTION-STATE function.

The operation of this type of connection is modeled on the standard procedures for the handling of modem signals and how connections are established and broken. This section deals with the handling of a connection once it has been established, in particular, establishing synchronization between the two DTE's using the connection (e.g., auto-baud) and the temporary loss of the "carrier" (CD) and "clear-to-send" (CTS) signals (standard definitions and abbreviations of modem signals are used in this section).

This section presents a state machine which

- o Reflects the states and state transitions involved in establishing and breaking a physical connection. (The purpose of these states is to allow terminal management to observe the state of the connection.

- o Describes the operation of the physical connection once the connection has been established.

When the physical terminal is connected to the server by a non-switched connection, the only transitions will be between CONNECTED-Active and CONNECTED-Sync-Wait (the modem signals DSR, CTS, and CD are assumed to always be true).

The CONNECTED-Sync-Wait state described below exists for establishing synchronization between the physical terminal and Foundation processes communicating with the terminal over the physical connection. Synchronization will include auto-baud (when the Auto-baud-detect characteristic is true), determining terminal type, and other activities necessary for the correct operation of the terminal over the physical connection.

The other states described below are related to modem operation.

There are four major states and some minor states for the CONNECTED and DISCONNECTED major states. The major state names are in all capital letters and the minor state names are lowercase except for the first letter of each word.

The physical connection states are described in Table 3-4.

Table 3-4 Physical Connection States

State	**Meaning**
DON'T-ANSWER	There is no connected physical terminal. A call will not be answered. ("Data terminal ready" is not true in this state.)
DISCONNECTED-Idle	There is no connected physical terminal. A call will be answered. ("Data terminal ready" is true.)
DISCONNECTED-Timeout	A period of time during which DTR is held off to ensure the connection is properly broken.
CONNECTING	A timeout initiated after DSR goes true before entering the CONNECTED state.

(continued on next page)

Table 3-4 (Cont.) Physical Connection States

State	Meaning
CONNECTED-Active	The physical terminal is ready for data transfer (in this state, a "clear to send" signal received by the server system and a "request to send" signal asserted by the server system are handled in an implementation-dependent manner.)
CONNECTED-Sync-Wait	A connection has been established but synchronization not yet established. All input characters are discarded.
CONNECTED-CD-Wait	A connection has been established, but "carrier" (CD) was temporarily lost. The connection will be broken if CD does not reappear within two seconds. All input characters are discarded.
CONNECTED-CTS-Wait	A connection has been established, but CTS was temporarily lost. The connection will be broken if CTS does not reappear within two seconds.

Table 3-5 summarizes the reasons for the transitions among these states.

Table 3-5 Physical Connection State Transition Reasons

Old State	New State	Transition Reason
DISCONNECTED-Idle	CONNECTING	The signal DSR is detected. Timeouts are started.
CONNECTING	DISCONNECTED-Timeout	The timeout started in the CONNECTING state expired before CD was detected.
CONNECTING	CONNECTED-Sync-Wait	A "carrier" signal was detected, and the Auto-baud-detect characteristic is true.
CONNECTING	CONNECTED-Active	A "carrier" signal was detected, and the Auto-baud-detect characteristic is false.
CONNECTED-Active	CONNECTED-CD-Wait	The "carrier" signal was lost. (A two-second timeout is started.)
CONNECTED-CD-Wait	CONNECTED-Active	"Carrier" returned before the timeout expired.
CONNECTED-CD-Wait	DISCONNECTED-Timeout	Timeout expired.

(continued on next page)

Table 3-5 (Cont.) Physical Connection State Transition Reasons

Old State	New State	Transition Reason
CONNECTED-Active	CONNECTED-Sync-Wait	Synchronization was lost (e.g., eight consecutive characters were received with a framing error and the Auto-baud-detect characteristic is true).
CONNECTED-Sync-Wait	CONNECTED-Active	The server has ascertained the baud rate.
CONNECTED-Sync-Wait	DISCONNECTED-Timeout	Fatal synchronization failure.
CONNECTED-Active	CONNECTED-CTS-Wait	The CTS signal was lost. (A two-second timeout is started.)
CONNECTED-CTS-Wait	CONNECTED-Active	CTS returned before the timeout expired.
CONNECTED-CTS-Wait	DISCONNECTED-Timeout	Timeout expired.
DISCONNECTED-Timeout	DISCONNECTED-Idle	Timeout expired.
any	DISCONNECTED-Timeout	The HANG function was executed at the terminal management interface, or the "data set ready" signal was lost.
any	DON'T-ANSWER	The DISABLE function was executed at the terminal management interface.
DON'T-ANSWER	DISCONNECTED-Idle	The ENABLE function was executed at the terminal management interface.

3.1.4 **Physical Terminal Interface** - The physical terminal interface exists between the server system and the physical terminal. (It is essentially the interface to a UART including modem control signals.)

This interface contains the following functions:

1. read a character from an entry device

2. write a character to a presentation device

3. set the character bit width

4. enable/disable parity

5. set parity type, if enabled

6. set input speed

7. set output speed

8. sense modem signals

9. set modem signals

READ-TERMINAL-CHAR (physical terminal id; return, character)

return	= One of the following:
	success - character returned
	failure - no character returned
	failure - line break
	failure - framing error
	failure - parity error
	failure - receiver overrun

An implementation of this specification may not distinguish some of these failures (e.g., "line break" may not be distinguished from "framing error"). Also, in the case where multiple errors could be observed simultaneously, this interface returns only a single error. The error precedence is as defined by the order given above except where a framing error is detected and a character is also received whose bits are not all zero -- this is to be interpreted as a "framing error" and not as a "line break".

NOTE

Some hardware doesn't specifically differentiate between "line break" and "framing error". Both are received as a "framing error" and the character received with the error condition is used to differentiate a "line break" (the line purposely being held in the SPACE state for greater than one character time) from a "framing error" (a received character whose stop bit was SPACE rather than MARK). Where this is the case, the character received with a "line break" will be null (all zeros) and the character received with a "framing error" will be non-null.

WRITE-TERMINAL-CHAR (physical terminal id, character; return)

return	= One of the following:
	success - character accepted for output
	failure - insufficient resources to queue another character

SET-CHAR-WIDTH (physical terminal id, char-width)

char-width	= 5, 6, 7, or 8.

ENABLE-PARITY (physical terminal id)

DISABLE-PARITY (physical terminal id)

SET-PARITY-TYPE (physical terminal id, parity-type)

> parity-type = One of the following:
>
> CLEAR (force parity bit 0)
>
> EVEN
>
> ODD
>
> SET (force parity bit 1)
>
> If parity is enabled, the parity-type specified by this function is used for parity checking and generation in the physical terminal (where these operations are supported by the physical terminal).

SET-INPUT-SPEED (physical terminal id, input-speed)

> input-speed = The bit/second input speed. A 16-bit value.

SET-OUTPUT-SPEED (physical terminal id, output-speed)

> output-speed = The bit/second output speed. A 16-bit value.

SENSE-MODEM (physical terminal id; data-set-ready, ring, carrier, clear-to-send)

> Each of these signals is returned as a Boolean value. The names are intended to convey the meaning of the signal as defined by EIA standard RS-232-C. The underlying hardware interface may be other than RS-232-C.

SET-MODEM (physical terminal id, data-terminal-ready, request-to-send)

> The number of stop bits associated with the terminal is calculated by the following algorithm. If the output speed is less that 300 bits/second, the number of stop bits is set to two. Otherwise, the number of stop bits is set to one.

3.1.5 **Internal Operation** - The logical terminal services module contains internal algorithms to enter and exit terminal management mode and to perform the processing implied by several of the logical terminal characteristics. A general description of this operation follows.

3.1.5.1 **Loss Notification** - If Loss-notification is TRUE and an attempt to place a character on either the logical terminal input queue or the management input queue fails because there is no room in the destination queue, then the Loss-notification-character is placed on the physical terminal output queue, if there is room on the output queue.

3.1.5.2 **Mode Switching** - Each physical terminal is in one of two modes: (1) normal mode or (2) terminal management mode. In the future, the terminal user may be able to switch from one mode to the other by using special keys (such as the VT100 SETUP key) that are not accessible to application programs. For the present, the terminal user switches from either of these modes to the other by entering Switch-character-1 followed by Switch-character-2 (see terminal characteristics, Section 3.1.1 and Appendix A.2).

To allow the user to pass this sequence of characters through without switching modes, the logical terminal services module converts two consecutive Switch-character-1's to a single Switch-character-1 and passes it through without switching modes. A Switch-character-1 followed by any character other than a Switch-character-2 are both passed through unmodified, without switching modes.

While a terminal is in terminal management mode, input characters are removed from the physical terminal input queue and placed on the terminal management input queue; characters written via the terminal management WRITE-CHAR function are placed on the output queue.

While a terminal is in normal mode, input characters are removed from the physical terminal input queue and placed on the logical terminal input queue (with the exceptions described below). Characters written via the normal mode access WRITE-CHAR function are placed on the output queue.

All output characters are subject to flow control and character expansion, as described below.

3.1.5.3 **Position Modeling, Character Expansion and Wrapping** - The logical terminal services module attempts to model the horizontal and vertical active position of the terminal for several reasons. It requires these values to expand form feeds and tabs and to do character wrapping. It also requires these values to provide the normal mode access WRITE-CHAR function.

The logical terminal services module maintains three state variables for each logical terminal: (1) "horizontal position", (2) "vertical position", and (3) "page stop position". The initial value of all these variables is 0. The variables are initialized whenever a new binding is formed and whenever a new mode is entered for the logical terminal. "Horizontal position" takes on values in the range 0 to Line-width minus 1. "Vertical position" and "page stop position" take on values in the range 0 to Page-length minus 1, and 0 to Stop-length minus 1, respectively.

In addition, the logical terminal services module is capable of expanding the HT, VT, and FF characters or attempting to track their position, assuming the physical terminal hardware is expanding them. This capability is selected via characteristics.

Each character placed on the output queue is examined to determine how these variables should be changed and what character expansions, if any, should be performed. This operation is summarized below. Unless noted otherwise, the page stop position is changed in the same way as the vertical position. (Character wrapping, which may affect horizontal and vertical position is also discussed.)

BS	A backspace decrements the horizontal position by 1. If the horizontal position is 0, it remains at zero.
HT	A horizontal tab is expanded to enough spaces to bring the horizontal position to the next multiple of 8. If the horizontal position becomes equal to <Line-width - 1> in this process, the expansion stops. (If wrapping is enabled, this last character causes a wrap.)
LF	A line feed increments the vertical position by 1 (mod Page-length).
VT	A vertical tab is expanded either to a form feed (described below) or to enough line feeds to bring the vertical position to the next multiple of 11 (mod Page-length). The page stop position is incremented by 1 for each increment in the vertical position.
FF	A form feed is expanded to enough line feeds to bring the vertical position to 0 (mod Page-length). The page stop position is incremented by 1 for each increment in the vertical position.
CR	A carriage return sets the horizontal position to 0.

Printing characters	These characters increment the horizontal position by 1 to a maximum value of <Line-width - 1>. (Wrapping may occur at this point.)
Other characters	These characters have no effect on the horizontal or vertical position.

If character wrapping is selected (via a characteristic), the following additional operation takes place. Each time the horizontal position is incremented, it is compared to Line-width. If they are equal, the horizontal and vertical positions are updated to reflect the insertion of a <carriage return, line feed> in the output preceding the character in question. If so requested by the Wrap characteristic, the logical terminal services module inserts these characters; otherwise, it assumes that the hardware has inserted the characters.

3.1.5.4 **Output Flow Control** - Output flow control is the control over output from the server to the physical terminal. It may operate at two levels: (1) as an "on/off" control and (2) as a "page stop" control. The former is selected by the Output-flow-control characteristic; the latter is selected by the Output-page-stop characteristic.

"On/off" flow control means that when the server receives an XOFF character from the terminal, it stops transmitting. When it receives an XON character, it starts transmitting again. Note that, for this function to provide a reasonable service as seen by the terminal user, the physical terminal output queue must be short and the logical terminal services module must have sufficient processing bandwidth. This processing is performed according to Digital's standard for defining the XOFF operation.

"Page stop" control means that the server stops output every time it has incremented the "page stop position" (discussed above) by the value of Stop-length. When it receives an XON character, it starts transmitting again. The "page position" variable, described above, is not set to 0 when the vertical position is set to 0. It is set to 0 only when transmission restarts.

If both flow control procedures are in effect at the same time, they are considered to be layered, with the "on/off" control below the "page stop" control. This means that if an XON is received, the lower state is first examined to see if transmission should restart. If the lower state is stopped, it is reenabled. (Transmission will not start, however, if the upper level is stopped.) If the lower state is enabled, then the XON is passed to the upper state and handled there. If that state is stopped, it is restarted. If that state is enabled, the XON is ignored.

The value of the page-stop-position variable can be reset to 0 by the mode module using the Reset Page-stop-position function, thus altering the point at which output will stop due to the page-stop-position variable reaching the Page-length limit. This function is used to differentiate between output from the host and input characters being echoed locally.

3.1.5.5 **Input Flow Control** - Input flow control is the control over input from the physical terminal to the server. It is controlled completely by an operation analogous to "on/off" output flow control, as described in Section 3.1.5.4. This operation is selected by a characteristic.

If either input queue crosses a threshold while a character is being added to it, and an XOFF character has not been sent to the terminal since the last XON was sent (as part of this algorithm), then an XOFF is placed on the physical terminal output queue. If either input queue crosses a threshold (not the same threshold described in the preceding paragraph) while a character is being removed from it, and an XON has not been sent to the terminal since the last XOFF was sent (as part of this algorithm), then an XON is placed on the physical terminal output queue.

3.2 Terminal Communication Services

The primary functions of the terminal communication services modules are to bind logical terminals to host portals, to synchronize the sharing of a binding among multiple pairs of mode access modules, and to carry mode access protocol messages between a host system and a server system.

3.2.1 **Logical Terminals and Portals** - A logical terminal has already been defined. For the purposes of the following connection management functional description, a logical terminal is an addressable collection of resources existing in a server system.

Similarly, a portal is an addressable collection of resources existing in a host system. The connection management functions allow a logical terminal and a portal to become logically connected. When this occurs, the portal and logical terminal are referred to as being bound, and the connection is referred to as a binding. When a logical terminal is bound to a portal, a mode access module in the host system may communicate with its counterpart in the server system via the binding.

This specification models logical terminals and portals as static resources; they are neither created nor destroyed by action of any of the modules described below. A future version of this specification may describe the dynamic creation and destruction of either or both.

3.2.2 **Version 1.0 Compatibility** - This version of this specification is compatible with the "DECnet Network Virtual Terminal Specification", Version 1.0. This compatibility is reflected in the interface described below. In particular, a logical terminal may become bound to a Version 1.0 portal, and a portal may become bound to a Version 1.0 logical terminal. Mode access modules detect this by examining the binding state of the logical terminal or portal. This state is BOUND when each of the parties is a Version 2.0 (or later) party; this state is BOUND-1 when one of the parties is a Version 1.0 party.

A logical terminal or portal in the BOUND-1 state has no associated mode state and is not affected by the mode management interface functions. It is assumed that a higher level module can properly manage such a logical terminal or portal.

3.2.3 **Host System Connection Management Functions** - The terminal communication services module in the host system provides the following connection management functions to higher level modules in the host:

1. read the set of portal id's
2. register a mode access module
3. read the portal binding state
4. bind to a terminal by name
5. bind to a communicating terminal
6. disconnect (unbind) a binding
7. close a binding

This specification does not attempt to define all the modules that might use the interface functions described below. The following information describes a typical binding scenario.

The terminal management module in a remote server system initiates a virtual circuit for the purpose of forming a binding. A login process in the host system to which the virtual circuit was directed scans the portals looking for one that has been associated with a newly-formed virtual circuit. (The description of portal binding states, a few paragraphs below, tells how this information is available.) The login process causes the portal to become bound by issuing a request to the terminal communication services module. Once the binding has been formed, the login process can prompt the user for login information by associating the portal with terminal service requests.

Alternatively, when a server system is initialized, some or all logical terminals may be configured to be bound to a particular application in a particular host in a particular mode. In this case, a module in the server system (unspecified here) and a module in the host system (also unspecified here) would form a binding, after which the host module would start the application and place the binding in the necessary mode.

Most of the functions described below are intuitively required to bind and unbind portals and logical terminals. The REGISTER-MODE function exists for the following reason.

It is assumed that a given binding may sequentially operate in several modes during its lifetime. This would be the case if for example; a binding was first placed in command mode for logging in; the user ran an application that placed the binding in forms mode; and then the host system placed the binding back in command mode when the application finished. This operational model assumes that a mode access module (in either the host or server system) may maintain active state information for a binding even when the binding is operating in a different mode.

The REGISTER-MODE and UNBIND functions with the connection management state machine ensure that all mode access modules participate in the process of unbinding a portal. This, in turn, allows a mode access module to properly initialize any state variables associated with a given portal, even if the module is not currently active on the portal.

The following two functions allow a mode access module to obtain the active set of portal id's. They are modeled for reasons similar to those for GET-FIRST-LOGICAL-TERMINAL-ID and GET-NEXT-LOGICAL-TERMINAL-ID, described in Section 3.1.2, Normal Mode Access Interface.

GET-FIRST-PORTAL-ID (;portal id)

portal id	= The value of the "first" portal id in the list of current portal id's maintained by the terminal communication services module.

GET-NEXT-PORTAL-ID (;return, portal id)

return	= One of the following:
	portal id returned
	no more portal id's
portal id	= The value of the "next" portal id in the list of current portal id's maintained by the terminal communication services module.

REGISTER-MODE (mode id; return)

mode id	= A mode access identifier (see Mode Management).
return	= One of the following: success - user registered failure - insufficient resources

This function is normally executed only once by each mode access module at system initialization. Each mode access module registered via this function must participate in the unbinding of a portal. See the connection management state description below.

READ-PORTAL-BINDING-STATE (portal id; state, source, name,
logical terminal id, reason)

state	= One of the following: UNBOUND REBIND-WAIT REQUESTING ALLOCATED BOUND BOUND-1 UNBINDING
source	= The node name of the server system attempting to form a binding (returned only if state = ALLOCATED)
name	= The name (defined by the DEFINE-NAME function described in Section 3.1.3) of the logical terminal attempting to form a binding. Returned only if state = ALLOCATED)
logterm id	= The logical terminal id of the bound logical terminal (returned only if state is BOUND)
reason	= Reason for entering UNBOUND state. One of the following: no reason no communication (returned only if UNBOUND state entered from REQUESTING state) requested terminal in use (returned only if UNBOUND state entered from REQUESTING state)

no resources (returned only if UNBOUND state entered from REQUESTING state)

requested terminal does not exist (returned only if UNBOUND state entered from REQUESTING state)

destination system not a server (returned only if UNBOUND state entered from REQUESTING state)

BIND-NAME (portal id, destination, name)

destination	= The node name of the server system to which the binding should be directed.
name	= The name (defined by the DEFINE-NAME function described in Section 3.1.3) of the logical terminal to which the binding is requested.

This function binds to a terminal by name. It is valid only if the portal is in the UNBOUND state.

BIND (portal id)

This function binds to a logical terminal which has communicated with the host for the purpose of forming a binding. It is valid only if the portal is in the ALLOCATED state.

UNBIND (portal id, mode id)

Each registered mode access module must issue this function for a portal to unbind. This function is valid only if the portal is in the REQUESTING, IN-USE, BOUND, BOUND-1, or UNBINDING state.

CLOSE (portal id)

This function requests the immediate change of the portal to the UNBOUND state.

Table 3-6 defines the meanings of the binding states of a portal.

Table 3-6 Portal Binding States

State	Meaning
UNBOUND	There is no connected logical terminal.
REQUESTING	The host system BIND or BIND-NAME function was requested; the portal is attempting to bind to the specified logical terminal.
ALLOCATED	A logical terminal is attempting to bind to this host via this portal (i.e., an incoming binding is in progress).
BOUND	A Version 2.0 logical terminal is connected to this portal.
BOUND-1	A Version 1.0 logical terminal is connected to this portal.
UNBINDING	The portal is attempting to make a transition to the UNBOUND state, but not all host system mode access modules have requested the UNBIND function.

Table 3-7 describes the reasons for the transitions among these states.

Table 3-7 Reasons for Portal Binding State Transitions

Old State	New State	Transition Reason
UNBOUND	REQUESTING	The host system BIND-NAME function was requested.
UNBOUND	ALLOCATED	The server system BIND function was requested at some point in the past.
REQUESTING	UNBOUND	One of the following: The destination system was not a server. The destination system had insufficient resources. The specified logical terminal did not exist. The specified logical terminal was already bound to a different portal.

(continued on next page)

Table 3-7 (Cont.): Reasons for Portal Binding State Transitions

Old State	New State	Transition Reason
REQUESTING	BOUND	The binding has been formed and the logical terminal is a Version 2.0 logical terminal.
REQUESTING	BOUND-1	The binding has been formed and the logical terminal is a Version 1.0 logical terminal.
ALLOCATED	REQUESTING	The host system BIND function was requested.
BOUND	UNBINDING	One of the following: The UNBIND function was requested at least once in the host system. The server system has failed. The underlying communication facilities have failed. The associated physical terminal has disconnected from the terminal system.
BOUND-1	UNBINDING	One of the following: The UNBIND function was requested at least once in the host system. The server system has failed. The underlying communication facilities have failed. The associated physical terminal has disconnected from the terminal system.
UNBINDING	UNBOUND	The host system UNBIND function was requested by the last registered mode access module.
any	UNBOUND	The host system CLOSE function was requested.

3.2.4 **Server System Connection Management Functions** - The terminal communication services module in the server system provides the following connection management functions for higher level modules in the system:

1. register a mode access module

2. read the logical terminal binding state

3. bind to a host by name

4. disconnect (unbind) a binding

5. close a binding

REGISTER-MODE (mode id; return)

mode id	= A mode access identifier (see Mode Management).
return	= One of the following:
	success - user registered
	failure - insufficient resources

This function is identical to its counterpart in a host system. In addition, it allows the server system to know when a mode change has been requested to a nonexistent mode (as described in Section 4.2.4, Server System).

READ-LOGICAL-TERMINAL-BINDING-STATE (logterm id; state, portal id, reason)

state	= One of the following:
	DISCONNECTED
	UNBOUND
	REQUESTING
	BOUND
	BOUND-1
	UNBINDING
portal id	= The portal id of the bound portal (returned only if state is BOUND or BOUND-1).

reason	= Reason for entering UNBOUND state. One of the following:
	no reason
	no communication (returned only if UNBOUND state entered from REQUESTING state)
	no resources (returned only if UNBOUND state entered from REQUESTING state)
	requested portal in use (returned only if UNBOUND state entered from REQUESTING state)
	requested portal does not exist (returned only if UNBOUND state entered from REQUESTING state)
	destination system not a host (returned only if UNBOUND state entered from REQUESTING state)
	protocol error

BIND (logterm id, destination)

destination	= The node name of the host system to which the binding should be directed.

This function is valid only if the logical terminal is in the UNBOUND state.

UNBIND (logterm id)

Each registered mode access module must issue this function for a logical terminal to unbind. This function is valid only if the logical terminal is in the REQUESTING, BOUND, BOUND-1, or UNBINDING state.

CLOSE (logterm id)

This function requests the immediate change of the logical terminal to the UNBOUND state.

Table 3-8 defines the meanings of the binding states of a logical terminal.

Table 3-8 Logical Terminal Binding States

State	Meaning
DISCONNECTED	There is no connected physical terminal associated with the logical terminal.
UNBOUND	There is no connected portal.
REQUESTING	The server system BIND function was requested; the logical terminal is attempting to bind to a host.
BOUND	A Version 2.0 portal is connected to this logical terminal.
BOUND-1	A Version 1.0 portal is connected to this logical terminal.
UNBINDING	The logical terminal is attempting to make a transition to the UNBOUND state, but not all server system mode access modules have requested the UNBIND function.

Table 3-9 describes the reasons for the transitions among these states.

Table 3-9 Reasons for Terminal Binding State Transitions

Old State	New State	Transition Reason
DISCONNECTED	UNBOUND	A physical terminal has become connected to the server system and associated with the logical terminal (the physical connection state is CONNECTED).
UNBOUND	DISCONNECTED	The associated physical terminal has disconnected from the server system (the physical connection state has left CONNECTED).
UNBOUND	REQUESTING	The server system BIND function was requested.
UNBOUND	BOUND	A host BIND-NAME function, specifying the corresponding logical terminal was requested at some point in the past.
REQUESTING	DISCONNECTED	The associated physical terminal has disconnected from the server system (the physical connection state has left CONNECTED).

(continued on next page)

Table 3-9 (Cont.): Reasons for Terminal Binding State Transitions

Old State	New State	Transition Reason
REQUESTING	UNBOUND	One of the following: The destination system was not a host. The destination system had insufficient resources. The specified portal did not exist. The specified portal was already bound to a different logical terminal.
REQUESTING	BOUND	The host system BIND function was requested at some point in the past and the portal was a Version 2.0 portal.
REQUESTING	BOUND-1	The host system BIND function was requested at some point in the past and the portal was a Version 1.0 portal.
BOUND	UNBINDING	One of the following: The UNBIND function was requested at least once in the terminal system. The host system has failed. The underlying communication facilities have failed. The associated physical terminal has disconnected from the terminal system.
BOUND-1	UNBINDING	One of the following: The UNBIND function was requested at least once in the terminal system. The host system has failed. The underlying communication facilities have failed.

(continued on next page)

Table 3-9 (Cont.): Reasons for Terminal Binding State Transitions

Old State	New State	Transition Reason
BOUND-1 (cont.)		The associated physical terminal has disconnected from the terminal system.
UNBINDING	DISCONNECTED	The server system UNBIND function was requested by the last registered mode access module and the associated physical terminal has disconnected from the server system.
UNBINDING	UNBOUND	The server system UNBIND function was requested by the last registered mode access module.
any	UNBOUND	The server system CLOSE function was requested.

3.2.5 **Host System Mode Management Functions** - The primary requirement of mode management functions (in both the host system and the server system) is to allow the orderly handing off of a binding from one pair of mode access modules to a second pair of mode access modules. This specification assumes that a mode change request is generally initiated in the host system by a mode access module. This is consistent with a model in which all terminal interactions are the result of explicit requests by a host module to send data, receive data, change control variables, etc. The exception to this occurs when a mode access module in a server system detects a protocol error in its protocol operation. In this case, a request to leave the current mode (or, equivalently, to enter no mode) may be made by such a mode access module (as described under Server System Mode Management Functions).

The host system interface to its terminal communication services module provides the following mode management functions:

1. enter a new mode

2. exit the current mode

3. confirm the remotely requested exit of the current mode

4. read the current mode state

ENTER-MODE (portal id, mode id)

This function requests either the entry of a mode when the portal was in the IN-NO-MODE state (which is the case just after a binding is formed) or the changing of modes from one mode to another (when the portal is in the IN-A-MODE state).

EXIT-MODE (portal id)

This function requests that the portal be taken out of any mode. It returns the portal to its state after the binding was initially formed. This function is valid only if the portal is in the IN-A-MODE state.

CONFIRM-EXIT (portal id)

This function confirms the exit of the old mode when the server system mode access module has requested such an exit.

READ-PORTAL-MODE-STATE (portal id; state, mode id, reason)

state	= One of the following: IN-NO-MODE EXITING IN-A-MODE ENTERING
mode id	= The mode that has been requested of the server system (if state = ENTERING) or the mode that the portal is in (if state = IN-A-MODE) or the mode that the portal was in (if state = EXITING).
reason	= The reason the state was entered (for some states). One of the following: host EXIT-MODE function executed (returned in the IN-NO-MODE state) requested mode doesn't exist (returned in the IN-NO-MODE state) server EXIT-MODE function executed (returned in the IN-NO-MODE and EXITING states)

Table 3-10 defines the meanings of the mode states of a portal.

Table 3-10 Host Portal Mode States

State	Meaning
IN-NO-MODE	There is no pair of mode access modules using the binding associated with the portal.
EXITING	The terminal mode access module that was using the asociated binding requested the EXIT-MODE function at some point in the past.
IN-A-MODE	A pair of mode access users is using the associated binding.
ENTERING	The host system ENTER-MODE function was requested.

Table 3-11 describes the reasons for the transitions among these states.

Table 3-11 Reasons for Portal Mode State Transitions

Old State	New State	Transition Reason
IN-NO-MODE	ENTERING	The host system ENTER-MODE function was requested.
EXITING	IN-NO-MODE	The host system CONFIRM-EXIT function was requested.
IN-A-MODE	IN-NO-MODE	The host system EXIT-MODE function was requested.
IN-A-MODE	EXITING	The server system EXIT-MODE function was requested at some point in the past.
IN-A-MODE	ENTERING	The host system ENTER-MODE function was requested.
ENTERING	IN-NO-MODE	The mode that was requested was not registered in the server system at the time the mode change request was processed there.
ENTERING	IN-A-MODE	The server system CONFIRM-ENTER-MODE function was requested at some point in the past.

3.2.6 **Server System Mode Management Functions** - The server system interface to its terminal communication services module provides the following mode management functions:

1. confirm a host request to enter a new mode

2. exit the current mode

3. confirm a host request to exit the current mode

4. read the current mode state

CONFIRM-ENTER-MODE (logterm id)

This function confirms the entry of a new mode. It is valid only if the logical terminal is in the ENTERING state.

EXIT-MODE (logterm id)

This function requests the exit of the current mode to no mode. This function is provided to allow a mode access module to take the binding out of any mode when it detects a protocol error in its protocol. This function is valid only when the logical terminal mode state is IN-A-MODE.

CONFIRM-EXIT-MODE (logterm id)

This function confirms the exit of an old mode. It is valid only if the logical terminal is in the CHANGING and EXITING states.

READ-LOGICAL-TERMINAL-MODE-STATE (logterm id; state, mode id, reason)

state	= One of the following: IN-NO-MODE CHANGING ENTERING IN-A-MODE EXITING
mode id	= The mode that has been requested of the logical terminal (if state = ENTERING), the mode that the logical terminal is in (if state = IN-A-MODE), or the mode that should exit (if state = either CHANGING or EXITING).

reason	=	The reason the state was entered (for some states). One of the following: server EXIT-MODE function executed (returned in the IN-NO-MODE state) host EXIT-MODE function executed (returned in the IN-NO-MODE and EXITING states)

Table 3-12 defines the meanings of the mode states of a logical terminal.

Table 3-12 Logical Terminal Mode States

State	**Meaning**
IN-NO-MODE	There is no pair of mode access modules using the binding associated with the logical terminal.
CHANGING	The host mode access module that was using the associated binding requested the ENTER-MODE function at some point in the past and the server CONFIRM-EXIT-MODE function has not been executed.
ENTERING	The host mode access module that was using the associated binding requested the ENTER-MODE function at some point in the past and, if the logical terminal had previously been in the IN-A-MODE state, the old mode module has requested the CONFIRM-EXIT-MODE function.
IN-A-MODE	A pair of mode access users is using the associated binding.
EXITING	The host EXIT-MODE function was executed at some point in the past.

Table 3-13 describes the reasons for the transitions among these states.

Table 3-13 Reasons for Terminal Mode State Transitions

Old State	New State	Transition Reason
IN-NO-MODE	ENTERING	The host system ENTER-MODE function was requested at some point in the past.
CHANGING	ENTERING	The server system CONFIRM-EXIT-MODE function was requested.
ENTERING	IN-A-MODE	The server system CONFIRM-ENTER-MODE function was requested.
IN-A-MODE	IN-NO-MODE	The server system EXIT-MODE function was requested.
IN-A-MODE	CHANGING	The host system ENTER-MODE function was requested at some point in the past.
IN-A-MODE	EXITING	The host system EXIT-MODE function was requested at some point in the past.
EXITING	IN-NO-MODE	The server system CONFIRM-EXIT-MODE function was requested.

3.2.7 **Data Transfer Functions** - If a host portal or a logical terminal is in the BOUND connection management state, a module layered above foundation services may send data to and receive data from its counterpart in the other system via the SEND-MESSAGE and RECEIVE-MESSAGE functions. Two different message types exist. COMMON messages are for communication between mode-independent layered modules (common terminal services), while MODE messages are for mode-dependent modules. The portal or logical terminal must be in the IN-A-MODE mode management state to send or receive MODE data.

- o send a message
- o receive a message

SEND-MESSAGE (message type, id, buffer; return)

message type	= Either COMMON or MODE.
id	= Either a portal id or a logical terminal id (depending on the system in which this function exists).
buffer	= A buffer containing a protocol message to transmit.

return	= One of the following: success - message queued internally for transmission failure - insufficient resources

RECEIVE-MESSAGE (message type, id, buffer; length, return)

message type	= Either COMMON or MODE.
id	= Either a portal id or a logical terminal id (depending on the system in which this function exists).
buffer	= An empty buffer to receive a protocol message.
length	= Length of received message in bytes if "return" is "success".
return	= One of the following: success - message returned in buffer failure - no message available

The synchronization between these functions, and the connection management and mode management functions, is described below.

- o Data between mode modules (not Common Terminal Service modules) may only be sent and received on a binding when it is in the BOUND connection management state and IN-A-MODE mode management state. Data between Common Terminal Service modules may be sent and received on a binding when its connection management state is BOUND.

- o An UNBIND function, ENTER-MODE function, or EXIT-MODE function may be thought of as being delivered synchronously with previously transmitted data. That is, if a module has sent several protocol messages and then requests the UNBIND, ENTER-MODE, or EXIT-MODE function, then all previously transmitted protocol messages are received in the remote system before the remote system perceives a transition of the binding to the UNBINDING connection management state or CHANGING or EXITING mode management states.

  Data received in the host system following a transition to the IN-A-MODE state are guaranteed to have been sent by the corresponding mode access module after the binding made a transition to the IN-A-MODE state in the server system.

- While a binding is either in a connection management state other than BOUND or in a mode management state other than IN-A-MODE, any received MODE type data that is internally queued or received via the network is discarded by the terminal communication services module. Similarly, COMMON type data is discarded if the connection management state is other than BOUND (the mode management state is not significant for COMMON type data).

3.2.8 **Virtual Circuit Services** - The terminal communication services modules require a virtual circuit service to communicate with each other. The virtual circuit interface functions they require are defined below.

CONNECT (node, object; return, port id)

node	= The name of the node to connect to.
object	= The identification of the module (in this case, the terminal communication services module in the host system) to connect to.
return	= One of the following: success - connection being requested failure - insufficient resources
port id	= An identification to be used for subsequent references to the virtual circuit.

SENSE-CONNECT (object; return, node, port id)

object	= The identification of the module (in this case, the terminal communication services module in the host system) that is requesting this function.
return	= One of the following: success - a connect request is pending failure - no connect request is pending
node	= The node name of the connection requestor (returned only on success).
port id	= An identification to be used for subsequent references to the virtual circuit (returned only on success).

ACCEPT-CONNECT (port id)

REJECT-CONNECT (port id)

SEND-DATA (port id, data; return)

return	= One of the following:
	success - data moved out of data buffer
	failure - insufficient resources

RECEIVE-DATA (port id, buffer; length, return)

length	= Length of received message in bytes if "return" is "success".
return	= One of the following:
	success - data placed in buffer
	failure - no data placed in buffer

DISCONNECT (port id)

SENSE-STATE (port id; state)

state	= One of the following:
	REQUESTING (CONNECT issued - no response received)
	RUNNING (ACCEPT function requested either locally or remotely)
	DISCONNECTED (DISCONNECT function requested either locally or remotely)

CLOSE (port id)

This function causes the port id to become undefined. It is normally used after the state is sensed to be DISCONNECTED.

4.0 **OPERATION**

The operation of the logical terminal services module and the distributed terminal communication services modules is described below. A complete operational specification should include a model implementation that is rigorous enough to answer most questions about how to implement this architecture in a product and that is abstract enough to be general. This specification lacks such a model implementation; however, it will be updated to include one in the future. For the present, the operational description takes the narrative form.

4.1 **Logical Terminal Services**

The description of the interface to these services defines them implicitly. Therefore, they are not elaborated on here.

4.2 **Terminal Communication Services**

The following description of the protocol messages exchanged between a pair of terminal communication services modules covers the general operation of the modules, the reasons for sending the messages, and the processing of received messages.

In the descriptions the term "host module" refers to the host system-resident terminal communication services module, and the term "server module" refers to the server system-resident terminal communication services module.

4.2.1 **Protocol Message Overview** - A pair of terminal communication services modules communicate via the terminal communication protocol. These modules are designed to use a virtual circuit communication service. This service is assumed to contain connect, disconnect, and data transfer functions. The message types in this protocol are described in Table 4-1. The "direction" column indicates if the message is sent from a host (H) to a server system (S), from a server system to a host, or from either one to the other.

Table 4-1 Terminal Communication Protocol Message Summary

Message	Direction	Definition
Bind Request	H ---> S	requests a binding; identifies version and type of sending system
Unbind	H <--> S	requests an unbinding
Bind Accept	H <--- S	accepts a bind request
Enter Mode	H ---> S	requests the entry of a new mode
Exit Mode	H <--> S	requests the exit of the current mode
Confirm Mode	H <--- S	confirms the entry of a new mode
No Mode	H <--- S	indicates that the requested mode is not available or confirms an exit mode request
Common Data	H <--> S	carries data for common terminal services
Mode Data	H <--> S	carries data for mode-dependent terminal services

4.2.2 **Protocol Errors** - A protocol error occurs when a protocol message is received which cannot be interpreted in a way that will ensure secure and synchronized operation. Unless otherwise specified, the occurrence of non-zero values in unused or reserved fixed length fields, and 1's in unused or reserved bits in bitmaps, can be ignored. All other errors constitute a protocol error -- see Section 4.3, Protocol Evolution for a statement of protocol compatibility.

A module detecting a protocol error disconnects the underlying virtual circuit.

4.2.3 **Connection Management Operational Overview** - In the terminal communication protocol, the host always sends the first message, regardless of which party originated the virtual circuit. This is done for compatibility with existing implementations of the DECnet network virtual terminal specification, Version 1.0. In these implementations, the server is responsible for supporting multiple protocols and for communicating with a given host in the host's native protocol.

After establishing a virtual circuit, the first message is a Bind Request message; hence, the host is always the party that "requests" the binding; the server "accepts" bindings.

In the following discussion of binding between the host system and the server system, the term "outgoing binding" refers to a binding originated by the system being discussed; "incoming binding" indicates a binding originated by the other system.

Host System

The host terminal communication services module saves a list of mode users (obtained from the REGISTER-MODE function) up to its internal resource capacity.

Outgoing Bindings:

When a host module receives a BIND-NAME request, it places the portal in the REQUESTING state and requests a virtual circuit connection to the server system specified in the request. It specifies the server module or a module that uses the terminal communication protocol described in this specification.

(This specification does not attempt to specify the network or network architecture that must be used for communication between a host module and a server module. However, when Phase III DECnet is used, the function above is accomplished by having the server module be object type 24, decimal.)

If the attempt to form the virtual circuit fails, the portal state is set to UNBOUND and the "reason" variable (returned on a READ-PORTAL-BINDING-STATE) is set to "no communication", "no resources", or "destination system not a server", as appropriate. (The latter return would be given if the virtual circuit service cannot locate a module that supports the server side of this protocol.)

If a virtual circuit is formed, the host module sends a Bind Request message. The host module waits for a response message. If the response is a Bind Accept message, the host module places the portal in the BOUND state. If the response is a version 1.0 Bind message, the host module places the logical terminal in the BOUND-1 state. If the response is an Unbind messsage, the host module sets the portal "reason" variable according to the REASON field from the message. If any other message is received, a protocol error has occurred and the host module places the portal in the UNBOUND state, sets the "reason" variable to "protocol error", and disconnects the virtual circuit. Note that use of reserved bits in the OPTIONS field is not treated as a protocol error.

Incoming Bindings:

The host module examines lower level communication ports, looking for a communication port that indicates an incoming virtual circuit either directed at the host module or for a module that uses the terminal communication protocol described in this specification. (If Phase III DECnet provides the communication functions, the host module is object type 42, decimal.)

When the host module detects an incoming virtual circuit connect request for itself, it accepts the connection if it has a portal in the UNBOUND state; otherwise it rejects the connection. The host module associates one of the UNBOUND portals with the virtual circuit and places the portal in the ALLOCATED state.

When a higher level module issues the BIND function, the host module places the portal in the REQUESTING state and sends a Bind Request message. The host module waits for a response message.

The remainder of the incoming binding operation is identical to the corresponding part of the outgoing binding operation, described above.

Unbinding:

If a higher level module requests the UNBIND function, the host module places the portal in the UNBINDING state and sends an Unbind message with reason "user unbind request". If the host module receives an Unbind message, detects a protocol error, or is notified by the underlying communication service that it has lost connection with the server system, it places the portal in the UNBINDING state. If this was due to a received Unbind message or a protocol error, the host module disconnects the virtual circuit.

The host module places a portal back in the UNBOUND state after it has received a number of UNBIND requests equal to the number of registered modes or after either receiving an Unbind message from the server or detecting the loss of the underlying virtual circuit.

NOTE

> An UNBIND with a valid reason code should be sent before breaking a virtual circuit. When a virtual circuit is dropped without an UNBIND, there is no way to differentiate between a successful shut-down and a failure condition of a type which tears the virtual circuit down without notice.

Closing:

When the CLOSE function is executed in the host, the host module closes the virtual circuit associated with the binding and places the portal in the UNBOUND state.

Server System

Outgoing Bindings:

When a server module receives a BIND request, it places the logical terminal in the REQUESTING state and requests a virtual circuit connection to the host specified in the request. It specifies the host module or a module that uses the terminal communication protocol described in this specification (see the discussion of host system in Section 4.2.4, Mode Management Operational Overview).

If the attempt to form the virtual circuit fails, the logical terminal state is set to UNBOUND and the "reason" variable (returned on a READ-LOGICAL-TERMINAL-BINDING-STATE) is set to "no communication", "no resources", or "destination system not a host", as appropriate. (The latter return would be given if the virtual circuit service cannot locate a module that supports the host side of this protocol.)

If a virtual circuit is formed, the server module waits to receive a Bind Request message from the host. If any other message is received, a protocol error has occurred. If a protocol error occurs, the server module disconnects the virtual circuit. Note that use of reserved values and bits in the OPSYS, SUPPORT, or OPTIONS fields are not treated as protocol errors.

If no protocol error occurs, the server module processes the VERSION value from the Bind Request message (described below), at the end of the formation of an incoming binding. On receiving a valid Bind Request from the host, the server returns a Bind Accept message. (The NAME field of the Bind Request message does not apply in the case of a server outgoing binding, because the server module has specified the logical terminal for the binding.)

Incoming Bindings:

The server module examines lower level communication ports, looking for a communication port that indicates an incoming virtual circuit either directed at the server module or for a module that uses the terminal communication protocol described in this specification.

When the server module detects an incoming virtual circuit connect request for itself, it accepts the connection if it has a logical terminal in the UNBOUND state; otherwise it rejects the connection.

The server module then waits to receive the first protocol message. This should be a Bind Request message. If any other message is received, a protocol error has occurred. If a protocol error occurs, the server module disconnects the virtual circuit. Note that use of reserved values and bits in the OPSYS, SUPPORT, or OPTIONS fields are not treated as a protocol error.

If no protocol error occurs, the server module examines the NAME field from the Bind Request message to ascertain if the requested terminal exists. This examination proceeds as follows. The name may be any length up to 40 characters. The server must support logical terminal names of at least 6 characters. The name from the NAME field of a Bind Request message matches a logical terminal name if they match character for character, assuming the shorter name is padded with blanks to the length of the longer name.

If the specified logical terminal does not exist, the server module sends an Unbind message, setting the REASON field to "selected logical terminal or portal does not exist". If the specified logical terminal exists but is not in the UNBOUND state, the server module sends an Unbind message, setting the REASON field to "selected logical terminal or portal is in use".

If the specified logical terminal is in the UNBOUND state, the server module associates the logical terminal with the virtual circuit on which the protocol messages have been received.

The server module then examines the VERSION value from the Bind Request message. If the host module's version is other than 1.0, the server module places the logical terminal in the BOUND state and sends a Bind Accept message.

If the host module's Version is 1.0, the server module places the logical terminal in the BOUND-1 state and sends a Bind message. This message is sent in conformance with the DECnet network virtual terminal specification, Version 1.0.

Unbinding:

Unbinding generally operates the same in the server system as in the host system. In addition, the logical terminal services module unbinds from a portal when it detects the disconnection of the corresponding physical terminal. In this case, it sends an Unbind message to the host with reason "terminal disconnected".

Closing:

When the CLOSE function is executed in the server, the server module closes the virtual circuit associated with the binding and places the logical terminal in the UNBOUND state.

4.2.4 **Mode Management Operational Overview - Host System**

When a host module receives an ENTER-MODE request, it places the portal in the ENTERING state and sends an Enter Mode message. It should eventually receive either a Confirm Mode or a No Mode message in response. If a Confirm Mode message is received, the host module places the portal in the IN-A-MODE state; if a No Mode message message is received, the host module places the portal in the IN-NO-MODE state. Until a response message is received, MODE data requested to be transmitted is discarded.

When a host module receives an EXIT-MODE request, it places the portal in the EXITING state and sends an Exit Mode message. It should eventually receive a No Mode message. If the host module receives an ENTER-MODE request before the EXIT-MODE request handling and protocol operation is complete, it operates as described in the preceding paragraph but must keep track of the order in which it expects to receive response messages.

If a host module receives an Exit Mode message while the portal is in the IN-A-MODE state, it places the portal in the EXITING state and sends no more MODE data until a higher level module executes a CONFIRM-EXIT request. It then places the portal in the IN-NO-MODE state.

Server System

When a server module receives an Enter Mode message for a logical terminal in the IN-A-MODE or IN-NO-MODE state, it examines the newly requested mode. If the mode is registered, the server module places the logical terminal in the CHANGING state. In this state, the current mode access module may continue to transmit, but it will receive no more messages. If the mode is not registered, the server module places the logical terminal in the EXITING state and sends a No Mode message.

In either the CHANGING or EXITING states, the current mode access module must execute the CONFIRM-EXIT function. When this occurs for the CHANGING state, the server module places the logical terminal in the ENTERING state for the newly requested mode. When this occurs for the EXITING state, the server module places the logical terminal in the IN-NO-MODE state.

If the logical terminal state is ENTERING and the new mode access module executes the CONFIRM-ENTER-MODE function, the server module places the logical terminal in the IN-A-MODE state and sends a Confirm Mode message.

When the server module receives an Exit Mode message, it places the logical terminal in the EXITING state and sends a No Mode message. It then operates as described in the case of receiving an Enter Mode message for a non-registered mode.

When the server module receives an EXIT-MODE request, it sends an Exit Mode message and places the logical terminal in the IN-NO-MODE state.

When a server module receives an Exit Mode message for a logical terminal in the IN-NO-MODE state, it sends a No Mode message.

4.2.5 **Data Transfer Operational Overview** - Data transfer operates the same in the host-system-to-terminal-system direction and the terminal-system-to-host-system direction. The SEND-MESSAGE and RECEIVE-MESSAGE functions are handled by passing the data in Data messages to the virtual circuit service.

4.3 **Protocol Evolution**

Extensibility is a requirement of this specification. The philosophy guiding the operation of a system in meeting this goal is the following. Compatibility is the responsibility of the implementation using the higher version of the protocol. Compatibility is guaranteed only across one major version number (that part of the version number before the decimal point) of the protocol and means the higher version must use a subset of its protocol that is consistent with correct operation of the implementation using the lower numbered version of the protocol. Undefined fields, subfields, and illegal values are protocol errors except for the Bind Request and Bind Accept messages where they are ignored (so the version numbers can be exchanged).

4.4 **Terminal Communication Protocol Messages**

The way in which a virtual circuit is used to carry these messages is described in Section 4.2.1, Protocol Message Overview. The format rules used for the messages described below are the same as used in Digital Network Architecture documents. Numbered bytes within a field are transmitted lowest to highest (byte 0, byte 1, etc.). Reserved values or bits in a bit map must be transmitted as zero. All messages have the following form:

MSGTYPE MSGDATA

```
     MSGTYPE (1) :  B    = Message type.  One of the following:

                           1  - Bind Request
                           2  - reserved
                           3  - Unbind
                           4  - Bind Accept
                           5  - Enter Mode
                           6  - Exit Mode
                           7  - Confirm Mode
                           8  - No Mode
                           9  - Common Data
                           10 - Mode Data
```

In the message descriptions, the entire message format, including MSGTYPE, is described for each message. For each message type, the value of MSGTYPE is considered a constant.

4.4.1 **Bind Request** - The format of this message is compatible with the Configuration message of the DECnet Network Virtual Terminal Specification, Version 1.0.

MSGTYPE VERSION OPSYS SUPPORT REVISION ID OPTIONS NAME

```
     MSGTYPE (1) :  C    = 1

     VERSION (3) :  B    = The version of the protocol as:

                           byte 0 - version number (2)

                           byte 1 - ECO number (0)

                           byte 2 - customer modification number (0)
```

OPSYS (2) : B = The operating system type of the sender. This field exists only for compatibility with the DECnet Network Virtual Terminal specification, Version 1.0. One of the following:

Value	Definition
0	unspecified
1	RT-11
2	RSTS/E
3	RSX-11S
4	RSX-11M
5	RSX-11D
6	IAS
7	VMS
8	TOPS-20
9	TOPS-10
10	OS-8
11	RTS-8
12	RSX-11M+

SUPPORT (2) : BM = The terminal communication protocol(s) supported by the sender (multiple bits may be set). This field exists only for compatibility with the DECnet Network Virtual Terminal specification, Version 1.0. Defined as:

Bit	Definition
0	RSTS/E DECnet homogeneous command terminal.
1	RSX family DECnet homogeneous command terminal.
2	VMS DECnet homogeneous command terminal.
3	TOPS-20 DECnet homogeneous command terminal.
4	Terminal communication protocol (this protocol).
5-15	Reserved.

REVISION (8) : A = The revision identification of the implementation sending this message; the contents of this field are implementation-dependent.

ID (2) : B = The logical terminal id or portal id associated by the sender with the binding.

OPTIONS (1) : BM = The options used by the sender, defined as:

Bit	Definition
0-7	Reserved.

NAME (I-40) : A = The name of the requested logical terminal (valid only if the host initiated the virtual circuit).

4.4.2 **Unbind** - The format of this message is compatible with the Disconnect Request message of the DECnet Network Virtual Terminal Specification, Version 1.0.

MSGTYPE REASON

MSGTYPE (1) : C = 2

REASON (2) : B = The reason for the unbinding. One of the following:

1 - incompatible versions

2 - no portal available

3 - user unbind request

4 - terminal disconnected

5 - selected logical terminal or portal is in use

6 - selected logical terminal or portal does not exist

7 - protocol error detected

4.4.3 **Bind Accept** -

MSGTYPE VERSION OPSYS REVISION ID OPTIONS

MSGTYPE (1) : C = 4

VERSION (3) : B = The version of the protocol as:

byte 0 - Version number (2)

byte 1 - ECO number (0)

byte 2 - customer modification number (0)

OPSYS (2) : B = The operating system type of the sender. This field exists only for compatibility with the DECnet Network Virtual Terminal specification, Version 1.0. One of the following:

Value	Definition
0	Unspecified
1	RT-11
2	RSTS/E
3	RSX-11S
4	RSX-11M
5	RSX-11D
6	IAS
7	VMS
8	TOPS-20
9	TOPS-10
10	OS-8
11	RTS-8
12	RSX-11M+

REVISION (8) : A = The revision identification of the implementation sending this message; the contents of this field are implementation-dependent.

ID (2) : B = The logical terminal id or portal id associated by the sender with the binding.

OPTIONS (1) : BM = The options used by the sender, defined as:

Bit	Definition
0-7	Reserved

4.4.4 **Enter Mode** -

MSGTYPE MODE

MSGTYPE (1) : C = 5

MODE (2) : B = The requested mode. (See Appendix A.1.)

4.4.5 **Exit Mode** -

MSGTYPE

MSGTYPE (1) : C = 6

4.4.6 **Confirm Mode** -

MSGTYPE

```
    MSGTYPE (1) :  C   = 7
```

4.4.7 **No Mode** -

MSGTYPE

```
    MSGTYPE (1) :  C   = 8
```

4.4.8 **Common Data** - The common data message is used to exchange protocol messages between mode-independent (common terminal service) modules layered above foundation services. The common data message supports blocking so more than one higher-level protocol message may be blocked into one common data message for transmission over the virtual circuit as a single unit. The CARRIED-MESSAGES field of the common data message is defined as a set of two repeating subfields where one pair of subfields is used for each protocol message blocked into the common data message.

MSGTYPE FILL CARRIED-MESSAGES

```
    MSGTYPE (1) :  C   = 9

    FILL (1) :  C      = 0

    CARRIED-MESSAGES   = A field containing the following repeating
                         subfields where there is one instance of the
                         subfield pair for each higher-level protocol
                         message being blocked into the common data
                         message. The subfield pairs are arranged
                         sequentially in the common data message
                         after the FILL field. The subfield pair is:

                         LENGTH  DATA

                         where:

                         LENGTH (2) : B  = The length of the
                                           following DATA subfield in
                                           bytes.

                         DATA       : A  = A single higher-level
                                           protocol message.
```

The total length of this message is determined from the length information provided by the virtual circuit service.

NOTE

To the layered modules using foundation services, blocked messages appear to be transmitted and received as separate messages.

4.4.9 **Mode Data** - The mode data message is used to exchange protocol messages between mode-dependent modules layered above foundation services. Like the common data message, the mode data message supports blocking of carried protocol messages.

MSGTYPE FILL CARRIED-MESSAGES

MSGTYPE (1) : C	= 10
FILL (1) : C	= 0
CARRIED-MESSAGES	= A field containing the following repeating subfields where there is one instance of the subfield pair for each higher-level protocol message being blocked into the mode data message. The subfield pairs are arranged sequentially in the mode data message after the FILL field. The subfield pair is:

LENGTH DATA

where:

LENGTH (2) : B	= The length of the following DATA subfield in bytes.
DATA : A	= A single higher-level protocol message.

The total length of this message is determined from the length information provided by the virtual circuit service.

4.5 Identifiers for Foundation-maintained Characteristics

Table 4-2, Logical Terminal Characteristics; and Table 4-3, Physical Terminal Characteristics, contain lists of characteristics with the corresponding identifiers and types as follows:

Type	Format	Definition
Boolean:	BM(1)	Low-order bit is the value (T = 1, F = 0)
Bit Map:	BM(x)	x specified individually
Integer:	B(2)	a 16-bit integer
String:	A	String of characters

Table 4-2 Logical Terminal Characteristics

Characteristic	Identifier	Type
MODE-WRITING-ALLOWED	1	Boolean
TERMINAL-ATTRIBUTES	2	Bit Map BM(2)
TERMINAL-TYPE	3	String
OUTPUT-FLOW-CONTROL	4	Boolean
OUTPUT-PAGE-STOP	5	Boolean
FLOW-CHARACTER-PASS-THROUGH	6	Boolean
INPUT-FLOW-CONTROL	7	Boolean
LOSS-NOTIFICATION	8	Boolean
LINE-WIDTH	9	Integer
PAGE-LENGTH	10	Integer
STOP-LENGTH	11	Integer
CR-FILL	12	Integer
LF-FILL	13	Integer
WRAP	14	Integer
HORIZONTAL-TAB	15	Integer
VERTICAL-TAB	16	Integer
FORM-FEED	17	Integer

Table 4-3 Physical Terminal Characteristics

Characteristic	Identifier	Type
INPUT-SPEED	1	Integer
OUTPUT-SPEED	2	Integer
CHARACTER-SIZE	3	Integer
PARITY-ENABLE	4	Boolean
PARITY-TYPE	5	Integer
MODEM-PRESENT	6	Boolean
AUTO-BAUD-DETECT	7	Boolean
MANAGEMENT-GUARANTEED	8	Boolean
SWITCH-CHARACTER-1	9	String
SWITCH-CHARACTER-2	10	String
EIGHT-BIT	11	Boolean
TERMINAL-MANAGEMENT-ENABLED	12	Boolean

APPENDIX A

Standards and Suggested Standards

The body of this specification does not define the standard way in which several of the specified capabilities may be used consistently through a network. This appendix contains standards and suggested standards for using some of these capabilities.

A.1 Standard Mode Values

To be useful, the values of the MODE field in an Enter Mode message require a networkwide standard definition. The standard definition of MODE is as follows:

MODE (2) : BM = DDDD MMMM MMMM MMMM

DDDD = Mode type having the following values:

Value	Definition
0	Corporate wide
1	Private
2	User

where

Corporate wide modes are DEC written modes which will be fairly widely accessible (i.e., implemented by most of DEC's systems) and must be approved by the Terminal Review Group.

Private modes are DEC written modes which are registered with the TRG but not necessarily approved by the TRG and are not as widely supported, perhaps only on a single system.

User Modes not known to the TRG for use by DEC customers, particularly, OEMs and universities, wanting to write their own modes -- the responsibility of coordinating these mode identification values rests with the users.

MMMMMMMMMMMM = The mode identification value within each of the above Mode Types.

Mode id values for the Corporate wide modes are:

Value	Mode
0	Invalid
1	Command mode

Mode id values for the private modes are:

Value	Mode
0	Invalid
-	To be defined

Mode id values for user modes are not defined here.

A.2 Suggested Setup/Normal Mode

Switching Characters The following definition of the Switch-character-1 and Switch-character-2 terminal characteristics is suggested.

```
Switch-character-1 = control-\
Switch-character-2 = carriage return
```

DECnet Digital Network Architecture
Network Virtual Terminal
Foundation Services
AA-DY89A-TK

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)__________

Name__________ Date__________

Organization__________

Street__________

City__________ State__________ Zip Code__________
or
Country

—Do Not Tear - Fold Here and Tape —

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE DOCUMENTATION
1925 ANDOVER STREET TWO/E07
TEWKSBURY, MASSACHUSETTS 01876

Do Not Tear - Fold Here and Tape

Cut Along Dotted Line