The data is transferred in ASCII form. The transfer byte size must be 8 bits. This type would be used for transfer of text files.
#
L - Local Byte
The manner in which data is to be transformed depends on the byte size for data transfer. This type is identical to the Image type for byte size which are integral multiples of or factors of the computer word length.
#
I - Image
on output the data is transformed from contiguous bits to bytes for transfer. On input, the data is transformed from bytes into bits, storing them contiguously independent of the byte size chosen for data transfer.
#
The following codes are not yet implemented:
#
E - EBCDIC
The data is transferred using the EBCDIC character code and 8-bit transfer byte size.
#
IMAGE
#
Declares IMAGE file type.
#
TENEX
#
Shorthand for IMAGE, BYTE 36.
#
ASCII
#
Shorthand for TYPE A, BYTE 8.
#
FORM format
#
The argument is a single ASCII character code specifying the format.
#
The following codes are assigned for format:
#
U - Unformatted
The representation type as specified is unaffected by any format transformations.
#
The following codes are not yet implemented:
#
P - Printfile
Data is transferred as either ASCII or EBCDIC type

in accordance with ASA (Fortran) vertical format control statements. The data is to be transferred in 8-bit bytes.

STRUCTURE structure-of-data

The argument is a single ASCII character code specifying file structure.

The following codes are assigned for structure-of-data:

F - File

The following codes are not yet implemented:

R - Record

MODE transmission-mode

The argument is a single ASCII character code specifying the data transfer rate modes.

The following codes are assigned for transfer modes:

S - Stream
The file is transmitted as a stream of bytes of the specified byte size. The EOF is signalled by closing the data connection. Any representation type and byte size may be used in the stream mode with file structure.

The following codes are not yet implemented:

B - Block
The file is transmitted as a series of data blocks preceded by one or more header bytes. The header bytes contain a count field, and descriptor code. The count field indicates the total length of the data block in bytes, thus marking the beginning of the next data block (there are no filler bits). The descriptor code defines last file block (EOF), last record block (EOR), restart marker, or suspect data. Record structures are allowed in this mode, and any representation type or byte size may be used.

T - Text
The file is ASCII text transmitted as a sequence of 8-bit bytes in the ASCII representation type, and optional Printfile format. Record structures

are allowed in this mode. The EOR and EOF are defined by the presence of special "TELNET-control" codes (most significant bit set to one) in the data stream. The EOR code is 192 (octal 300, hex C0). The EOF code is 193 (octal 301, hex C1). The byte size for transfer is 8 bits.

H - Hasp
The file is transmitted as a sequence of 8-bit bytes in the standard Hasp-compressed data format. This mode achieves considerable compression of data for print files. Record structures are allowed in the Hasp mode.

FTP Service Commands

GET REMOTE-FILE to LOCAL-FILE

The REMOTE-FILE is transferred from the host site to the user's site and is given the LOCAL-FILE name. Note: REMOTE-FILE and LOCAL-FILE use standard filename formats.

SEND LOCAL-FILE to REMOTE-FILE

The LOCAL-FILE is transferred from the user's site to the host site and is given the REMOTE-FILE name. Note: LOCAL-FILE and REMOTE-FILE use standard filename formats.

MULTIPLE GET/SEND

Only TENEX sites permit the use of "*" in the specification of filenames for GET and SEND. The standard "*" formats must be used.

APPEND LOCAL-FILE to REMOTE-FILE

The LOCAL-FILE is transferred from the user's site to the host site and appended to the REMOTE-FILE at the host site. Note: LOCAL-FILE and REMOTE-FILE use standard filename formats.

RENAME REMOTE-FILE to be NEW-REMOTE-FILE

The filename for REMOTE-FILE is changed to NEW-REMOTE-FILE. Note: REMOTE-FILE and NEW-REMOTE-FILE use standard filename formats.

DELETE REMOTE-FILE

This command deletes the REMOTE-FILE. Before the file is

actually deleted, the user is asked "Do you really want to delete? (Y or N)". Note: REMOTE-FILE uses standard filename formats.

DIRECTORY of REMOTE-FILE-GROUP

A list of the files in the REMOTE-FILE-GROUP will be typed out. (e.g., <SMITH>*.MAC, if remote site is a TENEX site)

STATUS of remote-system

Status information about the remote-system will be typed out.

MAIL <FILE> to REMOTE-USER

Sends LOCAL-FILE to mailbox at remote site.

HELP

Types summary of FTP commands.

Miscellaneous Commands

VERBOSE

Types out all comments in long form.

BRIEF

Types out all comments in short form.

QUOTE arbitrary-FTP-line

Sends arbitrary-FTP-line to remote-site without interpretation.

STATISTICS

Turns on typeout of timing statistics.

NOSTATISTICS

Turns off typeout of timing statistics.

FTP Control Characters

Control-G

Type BELL (^G) to abort a file transfer and return to command level.

```
# Control-O
#
#       Type ^O to clear typeout buffer.
#
# Control-V
#
#       Use ^V to quote characters in LOGIN.
#
#                           Example of FTP Use
#
#       @FTP                                    ;call in the subsystem
#       *BBN                                    ;connect to host BBN
#       *LOG SMITH SECRET 12345                 ;declare name, password,
#                                               ;account
#                                               ; the password will not be
#                                               ; echoed.
#       *DIR *.MAC
#       (to local file) TTY: [confirm]          ;get a partial directory
#                                               ; listing
#       *GET PROGRAM.MAC                        ;must end with carriage
#                                               ; return
#       (to local file> <esc>PROGRAM.MAC        ;escape causes same name
#                                               ; to be used
#       *DISCON                                 ;break the network
#                                               ; connections
#       *QUIT
#       @
```

INDEX

Commands are given in capital letters.

```
#                                HOSTAT
#
# HOSTAT obtains the network site status information maintained by
# the network survey site (MIT-DMS as of Dec. 1, 1974). It then
# types this out in columnated form, grouped by status. If the
# site name of a particular site is not known to the local TENEX,
# the octal site address will be given instead.
#
# The relevant status categories are:
#
# Logging:              The survey site was able to complete the ICP to
#                       this site's logger socket. (code 5)
#
# Refusing:             This site responded to an RFC to the logging
#                       socket, but did not complete the ICP. (code 4)
#
# Not Responding:       This site did not respond to an RFC to the
#                       logging socket. (code 3)
#
# No NCP:               This site's NCP did not respond at all. (code 2)
#
# Dead:                 This site is disconnected from the network.
#                       (code 1)
#
# TIPs up:              These sites are TIPs which responded to a probe
#                       from the survey site. (code 4)
#
# TIPs Down:            These sites are TIPs which are disconnected from
#                       the network. (code 1)
#
# Status Code Unknown:  The status code given by the survey site
#                       is not known.
#
# Unable to Poll:       The survey site was unable to poll this site
#                       successfully. (code 7)
#
# For more information about the survey service and codes see RFC
# 530 (A report on Survey Project by Abhay Bushan) and the
# subsequent note on this RFC (NIC Journal 20248).
#
# If the survey site is down, HOSTAT will call NETSTAT to provide
# the status information in the local TENEX's internal tables.
# Note that this information is updated somewhat randomly (ie.
# only when interactions occur or are attempted with the particular
# site involved).
#
# To use HOSTAT:
#
# @HOSTAT
# Getting survey from MIT-DMS... OK.
# Survey of 29-NOV-73 1:57PM-EST
```

```
#  Logging:
# SRI-ARC  CASE-10 USC-44  UCLA-CCN LL-TX-2 USC-ISI   MIT-AI
# RAND-RCC AMES-67 UCSD-CC MIT-DMS  CMU-10A UCLA-CCBS MIT-ML
# SU-AI
#
#  Not responding:
# LL-67 PARC-MAXC
#
#  Dead:
# UCLA-NMC   MIT-MULTICS CMU-10B  CCA-TENEX SRI-AI    BBN-TENEXB
# UCSB-MOD75 SDC-LAB     I4-TENEX UKICS-360 BBN-TENEX 243
# UTAH-10    HARV-10
#
#  TIPs Up:
# UTAH-TIP  RADC-TIP USC-TIP  SDAC-TIP   CCA-TIP  NCC-TIP
# AMES-TIP  NBS-TIP  GWC-TIP  ARPA-TIP   FNWC-TIP NORSAR-TIP
# MITRE-TIP ETAC-TIP DOCB-TIP BBN-TESTIP RML-TIP  UKICS-TIP
```

NETED, AN ARPA NETWORK COMMON EDITOR
#
#
NETED is a context editor which will present "the same" user
interface on many ARPA Network Hosts. The design is a moderate
extension of an extremely scaled-down editor derived from the
"ed" family of editors on CTSS (which was one of the very first
general purpose time-sharing systems). No claims are advanced
for its elegance or modernity; it is meant to be "a" common
editor for the Network, not "the" common editor -- but the
underlying design has stood the test of time. It was designed
and initially implemented by volunteers from the Network User
Interest Group (USING), who stand ready to supply both source and
object code to all Hosts. The major needs which it is intended
to satisfy are those of learnability and wide-spread
applicability. That is, it is a deliberately "un-fancy" editor,
so that new users can pick it up quickly; and it is meant to be
available on as many Network Hosts as possible, so that a single
investment in learning an editor will have a large payoff to
those who use more than one Host.
#
Although it is believed that the user interface is as similar as
possible from implementation to implementation, there are two
levels on which differences will be found. The more substantive
level is that of unavoidable system differences, such as the
availability or unavailability of "type-ahead", the specification
of characters for editing within a line, limits on line length
and file size, availability of both alphabetic cases, and the
like. On this level, the implementers' intent was to allow the
user to create a file which "makes sense" to the system on which
it is being created, rather than to attempt to constrain
ourselves to an unusable but identical "virtual" file format. So
there will be some unstated assumptions which differ from system
to system, which users will have to pick up as they go along --
but we have tried hard to keep them to a minimum.
#
The second level of differences is "cosmetic". That is, we have
not striven to make the form of all messages from the editor
identical; but we have striven to make their sense identical.
Because implementation strategies differ, and because the
implementers were volunteers, it did not seem appropriate to
attempt to "lock-step" in this area. An attempt has been made to
keep "normal" responses close enough to allow for the possibility
of invoking the editor from a program, but "abnormal" responses
have not been given the same attention.
#
If, in actual practice, cosmetic variations prove to be a major
nuisance to users -- or if users wish to offer comments on other
aspects of NETED -- the USING NETED committee can be reached by
Network mail through the Network Information Center's "Journal"
mechanism (address: yourident/neted), or through the Network

Feedback mechanism. Also, the Network Technical Liaisons at
Hosts running NETED implementations should be able to put users
in touch with the committee.
#
(A final introductory point: on systems which do permit
"type-ahead" the policy is to ignore any pending requests when an
abnormal condition is encountered while processing the "current"
request. This strategy is employed in order to prevent the
possibility of erroneously processing a request which was in some
sense contingent upon the one which did not complete normally.
Taking advantage of type-ahead, then, requires getting the "feel"
of the particular implementation being used. The editor is,
however, quite usable even without type-ahead -- and, as noted
above, not all implementations even offer it. Note also that
"prompting" is available, so that the user can be certain the
previous request is complete before typing a new one.)
#
"Modes"
As is typical of "context editors", the NETED command is used
both for creating new files and for altering already existing
files -- where "files" are named collections of character encoded
data in the storage hierarchy of a time-sharing system.
Consequently, NETED operates in two distinct "modes" -- called
"input mode" and "edit mode".
#
When NETED is used to create a file (that is, when it is invoked
from command level with an argument which specifies the name of a
file which does not already exist in the user's "working
directory"), it is automatically in input mode. It will announce
this fact by outputting a message along the lines of "File
soandso not found. Input." Until you take explicit action to
leave input mode, everything you type will go into the specified
file. (Actually, it goes into a "working copy" of the file, and
into the real file only when you indicate a desire to have that
happen.) Lines consisting of only a new-line are permitted.
Blanks at the end of input lines are handled differently by
different operating systems, and there is no general policy in
NETED on this topic (that is, some systems strip trailing blanks
"automatically" in their input/output subsystems -- before a
NETED implementation sees them). To leave input mode, type a
line consisting of only a period and the appropriate new-line
character: ".<NL>", where <NL> is whatever it takes to cause a
Telnet New-Line to be generated from your terminal.
#
After leaving input mode, you are in edit mode. Here, you may
issue various "requests" which will allow you to alter the
contents of the (working) file, re-enter input mode if you wish,
and eventually cause the file to be stored. Note that edit mode
is entered automatically if the argument you supplied to NETED
specified an existing file.

Regardless of how it was entered, being in edit mode is confirmed
by NETED's outputting a message of the form "Edit." Editing is
performed relative to a (conceptual) pointer which specifies the
current line, and many requests pertain to either moving the
pointer or changing the contents of the current line. (When edit
mode is entered from input mode, the pointer is at the last line
input; when entered from command level, the pointer is at the
"top" of the file.)
#
(Note that in the examples which follow although the command's
name and the requests are shown in lower case for typing
convenience, upper case also works; indeed, on many systems only
upper case is available.)
#
Invocation of the Command
Some time-sharing systems do not easily allow the passing of
arguments (or "parameters") to a program being invoked as a
command, while to others this is a natural approach. Because
NETED is specifically intended to be usable on all ARPA Network
Server Hosts, it has been given two methods of invocation in
light of these two styles of argument treatment:
1) neted
2) neted filename
#
Method 1) will work for all NETED implementations; method 2)
actually works for most implementations, but not all. Therefore,
it is advisable to invoke NETED on a system new to you via method
1) the first time you use it, and by whichever method you prefer
of the ones available subsequently. Note that in method 1), the
command will immediately ask you to enter a file name. As
indicated above, you will be in edit mode if the specified file
exists, and in input mode if not. A message will be output
indicating which mode has been entered.
#
Requests
NETED'S edit mode requests follow, in an order intended to be
helpful. Two important reminders: the requests may only be
issued from edit mode, and each one "is a line" (i.e., terminates
in a newline / carriage return / linefeed as appropriate to the
User Telnet being employed). Syntax Note: if the request takes
an argument, there must be at least one space (blank) between the
request's name and the argument. In certain of the requests (in
which the argument is a string of characters to be located or
inserted) leading blanks "in" the argument may be desirable;
thus, in the "l", "i", and "r" requests if more than one blank
has been used to separate the request name from its argument, the
additional blanks are significant, while in the rest of the
requests which take arguments such leading blanks are permitted
but ignored (see also Examples, below).
#

1. n m
For unsigned m, the n(ext) request causes the pointer to be moved "down" m lines. If m is negative, the pointer is moved "up" m lines. If m is not specified, the pointer is moved one line. The line is output unless printing has been suppressed by the "v" request (14.). If the end of the file is reached, an "End of file reached by n m" message is output by NETED, and the pointer is left "after" the last line.
#
2. l string
The l(ocate) request causes the pointer to be moved to the next line containing the character string string (which may contain leading and/or internal blanks); the line is output, unless printing is suppressed. If no match is found, a message of the form "Top of file reached by l string" will be output (and the pointer will have returned to the top of the file). The search will not wrap around the end of the file; however, if the string was above the starting position of the pointer, a repetition of the locate request will find it, in view of the fact that the pointer has been moved to the top of the file. To find any occurrence of the string -- rather than the next occurrence -- it is necessary to move the pointer to the top of the file before doing the locate (see following request).
#
3. t
Move the pointer to the top of the file (actually to a position "above" the current first line of the contents).
#
4. b
Move the pointer to the bottom of the file and enter input mode. "Input." will be printed when the request has completed.
#
5. .
Leave the pointer where it is and enter input mode. (First new line goes after current old line.) "Input." will be printed when the request has completed. Note that you remain in input mode until a line consisting of only ".<NL>" is given, as discussed above.
#
6. i string
The i(nsert) request causes a line consisting of string (which may contain leading and/or internal blanks) to be inserted after the current line. The pointer is moved to the new line. Edit mode is not left. If no string is given, a blank line will be inserted into the working file.
#
7. r string
The r(eplace) request causes a line consisting of string (which may contain leading and/or internal blanks) to replace the current line.

```
# 8. p  m
# The p(rint) request causes the current line and the succeeding m
#  -
# 1  lines  to  be output.  If m is not specified, only the current
# line will be output.  End of file considerations are the same  as
# with "n".  The pointer is moved to the final line printed.
#
# 9. c  /s1/s2/  m  g
# The  c(hange)  request  is quite powerful, although perhaps a bit
# complex to new users.  (Several examples of  its  use  are  given
# below.)   In  the line being pointed at, the string of characters
# s1 is replaced by the string of characters s2.  If s1 is void, s2
# will be inserted at the beginning of the line; if s2 is void,  s1
# will  be  deleted  from  the  line.  Any printing character not
# appearing within either character string may be used in place  of
# the  slash  (/)  as a delimiter.  If a number, m, is present, the
# request will affect m lines, starting with the one being  pointed
# at.   All  lines  in  which  a change was made are output, unless
# printing has been suppressed by the "v" request.  The pointer  is
# left  at  the  last  line  scanned.   If the letter "g" is absent
# (after the final delimiter)  only  the  first  occurrence  of  s1
# within a line will be changed.  If "g" (for "global") is present,
# all  occurrences  of s1 within a line will be changed.  (If s1 is
# void, "g" has no effect.)  Note well: blanks in both strings  are
# significant   and   must   be   counted  exactly.  End  of  file
# considerations are the same as with "n".
#
# 10. d  m
# The d(elete) request causes m lines, including the  current  one,
# to  be  deleted  from  the working copy of the file.  If m is not
# specified, only the current line  is  deleted.  The  pointer  is
# moved  to the immediately previous undeleted line (that is, it is
# moved "up"), except for the case where all the lines in the  file
# have  been  deleted,  in  which  case  the pointer will have been
# backed up to the top of the file.
#
# 11. w  filename
# Write out the working copy into the storage hierarchy  under  the
# name  filename  if this argument is present, or with the original
# name if the argument is not given, and remain in NETED.  See also
# discussion of "quit"  request  (18.).   A  message  of  the  form
# "filename  written."  is  output  when the request has completed.
# Pointer position is preserved.
#                .
# 12. save
# Write out the working copy into the storage  hierarchy  and  exit
# from NETED.
#
# 13. v
# The  v(erify)  request  reverses  the  setting  of  the  internal
# variable which governs printing of lines reached or  affected  by
```

several requests ("c", "l", and "n" -- but not "p"). It is
typically employed to suppress the large quantities of output
which result from multi-line changes, although experienced users
sometimes employ it merely to avoid waiting for responses when
they are confident they know what they're doing. Default is "on"
(i.e., such lines will be printed unless the "v" request is
given, and will again be printed if the "v" request is given
again).
#
14. *
Reverse the setting of the internal variable which governs
printing of "*" as a "prompt" when ready for a new request or
line of input. Because the editor is expected to be used heavily
by new users, the default for this variable is "on" in order to
offer initial reassurance that the editor "is there". That is,
prompting will occur unless the "*" request is given, and will be
resumed if the request is given again.
#
15. m filename
The m(erge) request causes a copy of the contents of the already
existing file designated by filename to be inserted into the
working file, beginning immediately after the current pointer
position. The pointer will be moved to the final line of the
inserted material, and a message of the form "filename merged."
will be output, when the request has completed.
#
16. h
The h(elp) request actuates a per-implementation help mechanism,
which ranges from a simple statement that the present
implementation has no known deviations from the standard (as
expressed in this document), through a list of local extensions
and (possibly) idiosyncrasies, to a full-blown tutorial on the
use of NETED. Each implementation will include an announcement
of its "erase" and "kill" characters (which, respectively, cause
one or more immediately previous characters or "print positions"
within a line to be erased, or cause the whole line up to that
point to be discarded); see also Note c), below.
#
17. ?
This request causes a list of the available requests to be
output.
#
18. quit
The quit request causes immediate exit from NETED; no writing out
of files occurs as a result of this request, although any
previous writing is unaffected by it. (If, for example, you wish
to alter a given file but preserve a copy of the original intact,
you would edit it, write the new version out under another name,
then "quit". To change the original, on the other hand, you
would exit either by "save" or by the sequence "w", "quit".)
N.B., work can be lost by improper use of "quit".

Notes
a) The requirement that "save" and "quit" be fully spelled out
is based on a concern that these two particularly powerful
functions not be invoked by the accidental mistyping of a single
letter.
#
b) At both the "top" and the "end" of the file, the editor is
dealing with a state rather than with a line. Messages of the
form "Top of file reached by:" and "End of file reached by:"
convey this idea when requests cause the editor to reach these
states. It is specifically declared to be an invalid operation
to attempt to r(eplace) or c(hange) when in such a state, and
implementations may vary in their responses to p(rint). (The
intended responses are "<Top of file.>" and "<End of file.>", but
by some implementation strategies the "reached by" type messages
or explicit error messages may be output instead.) In actual
practice, matters are less confusing than the above might seem to
indicate, for the only operation which one typically desires to
perform when at the top of the file is an insertion of some sort,
which works. (A change to the first line of the file's contents
is effected by "t","n", "c...".)
#
c) There are no defined "erase" and "kill" characters in NETED,
for two reasons: the user has available to him the editing
characters of his User Telnet (and/or the generic ones of the
Telnet Protocol) before transmission and the Server's after
transmission, so there is no real need for such a facility; and
the proliferation of special purpose characters on a per-command
basis is generally accounted an undesirable design practice.
(Use the "h" request to determine a given implementation's "line
editing" characters.)
#
d) On some systems, the system itself will emit a prompt
character whenever a call is made to read from the user's
terminal; the "*" request cannot, of course, turn off prompting
at that level.
#
e) By convention, per-implementation requests' names will begin
with an "x". (Use the "h" request to determine whether a given
implementation offers any such requests.)
#
Examples
In the following, the prompting "*'s" have been omitted because
they lead to considerable clutter and make the examples much
harder to read and follow. Please note that in actual use, the
editor will output a "*" when it is ready for a line of input (in
both edit mode and input mode). Those who prefer not to be
prompted in this fashion can turn prompting off with the "*"
request. (Remember that if you are creating a new file you must
shift to edit mode first. It is felt by the designers that this
slight inconvenience is offset by the greater convenience

```
# inexperienced  users  are afforded by having prompting be "on" by
# default, but we are sorry that the binary  choice  forces  us  to
# inconvenience anyone.)
#
# 1. Input and edit modes
# Assuming  that there is no file named "sample" in your directory,
# the command
#           neted sample
#
# would cause the response
#           File not found.
#           Input.
#
# Typing the following
#           This is line 1.
#           This is line 2.
#           This is line 3.
#           .
#
# would cause the three lines of text to be placed in  the  working
# copy  of the file, and generate the response (because of the mode
# change request ".")
#           Edit.
#
# The following sequence would write a copy  of  the  working  copy
# out, move the conceptual pointer to the top of the file, insert a
# line  there,  then re-enter input mode at the bottom of the file:
#           w
#           sample written.                          (response)
#           t
#           i This is line Ø.
#
#           b
#
# (Response after the "b" request is "Input.".)   Now  we  add  two
# lines at the bottom and return to edit mode:
#           This is line 4.
#           This is line 5.
#           .
#
# (Response is "Edit.")  At this point,
#           save
#
# will  write  out the (six-line) file and return to command level.
# Note that had it been desired to input more than one line at  the
# top  of the file (or elsewhere in the file) the "." request could
# have been used conveniently to enter input mode.
#
# 2. Pointer-moving requests
# Continuing with the file "sample", the following would leave  the
# pointer at the final line:
```

```
#           neted sample
#           Edit.                   (response)
#           n 6
#
# Note  that the argument to the "n" request is "6" rather than "5"
# because the top of the file is a null line rather than the  first
# line  of  the  contents,  in order to facilitate insertion of new
# material at the top of the file.  (If you had done  an  immediate
# "p"  request  after  entering  edit  mode from command level, the
# response would have been "<Top of file.>")  An alternate  way  of
# moving the pointer to the last line (instead of "n 6") is
#           1 5
#           This is line 5.                 (response)
#
# This  latter  method,  usually known as "locating by context," is
# the more common.  At this point,
#           n -2
# would cause the response
#           This is line 3.
#
# As noted above, "t" moves the pointer to the top of the file, and
# "b" moves it to the bottom (and enters input mode).
#
# 3. Changing existing lines
# Assume the pointer is still located at "This is line 3."
#           c /is/was/
#
# would result in
#           Thwas is line 3.
#
# Ah well.  Blanks are significant.  To fix the mess  and  do  what
# was intended:
#           c /was/is/
#           This is line 3.               (response)
#           c / is/ was/
#           This was line 3.              (response)
#
# To change all instances of a character string on a given line:
#           c /i/x/ g
#           Thxs was lxne 3.              (response)
#
# (Note  the  space  before the "g".)  An easy way to fix that line
# would be
#           r This is line 3.
#
# which simply replaces the current line.  ("c /x/i/ g" would  also
# work, of course.)
#
# The  following  request  (the  pointer  is not changed by the "r"
# request)
#           c /line/entry/ 2
```

```
# would result in the response
#           This is entry 3.
#           This is entry 4.
#
# with the pointer now at "This is entry 4."
#
# To append to the beginning of a line,
#           c //tag:/
#           tag:This is entry 4.                    (response)
#
# And to remove a string from a line,
#           c /tag://
#           This is entry 4.              (response)
#
# Note that "/" need not  be  used  as  the  delimiter.  I.e.,  "c
# xtag:xx" would also have worked in the last instance.
#
# 4. Miscellaneous requests
# Still using "sample" consider the following:
#           t
#           n
#           This is line Ø.               (response)
#           d 2
#           Top of file reached by "d 2"            (response)
#           i      This is the beginning.
#           c /in/inn/
#                This is the beginning.             (response)
#           l 3
#           This is entry 3.              (response)
#           d 99
#           End of file reached by "d 99"           (response)
#           .
#           Input.                (response)
#                This is the end.
#           .
#           Edit.                 (response)
#           t
#           p 99
#           <Top of file.>                (response)
#                This is the beginning.             (response)
#           This is line 2.               (response)
#                This is the end.                   (response)
#           End of file reached by "p 99" (response)
#
# Note  that  the  first  "d" request took care of the lines ending
# with "Ø." and "1."  and the second took care of "3." through "5."
# The response to the first illustrates what happens to the pointer
# after a "d" request.  The insertion of "This  is  the beginning."
# shows  the handling of leading blanks in edit mode; the insertion
# of "This is the end." shows the handling  of  leading  blanks  in
# input  mode.  The "." after the "d 99" could also have been a "b"
```

or an "i" request. A "save" request at this point would leave
you with a file containing only the three text lines which were
printed in response to the "p 99".
#
No attempt has been made here to offer examples of all possible
requests, in the belief that the foregoing examples convey the
"feel" of the editor and the descriptions of the other individual
requests should be sufficient for the user to learn from, once
the overall mechanism has been learned. Special note should be
taken of the "h" request, however, as in many implementations the
response to it will furnish important additional information.
#

NETSTAT

To obtain information on the status of the network as seen from TENEX:

@NETSTAT%

NETSTAT responds with:

*

and awaits a command.

(Note: Throughout this description of NETSTAT "%" indicates where carriage return is typed.)

There are two types of information which can be requested:

1) A list of the network sites which the local host considers to be up.
2) A list of network connections with the local host.

The following commands select the type of information to be given:

*ALL	All sites and all connections.
*CONNECTIONS	All connections.
*HOSTS	All hosts (i.e. not TIPS).
*SITES	All sites.
*SPECIFIC CONNECTIONS	Only connections of a specified type.

A blank line terminates commands and starts the type out.

If no command has been given then ALL is assumed.
eg:

@NETSTAT%
*%

Will list all sites and all connections.

SITES:

The site names are typed in tabular form with (octal) site addresses given for sites whose names are not known to TENEX.
eg:

@NETSTAT%

*SITES %
*%

THE FOLLOWING ARE UP:

```
UCLA-NMC    BBN-TENEXB SU-AI     MITRE-TIP USC-ISI DOCB-TIP
UCLA-CCN    MIT-DMS    ILL-ANTS  RADC-TIP  Ø27     SAAC-TIP
SRI-ARC     MIT-AI     I4-TENEX  NBS-TIP   USC-TIP ARPA-TIP
UCSB-MOD75  LL-67      AMES-67   ETAC-TIP  GWC-TIP Ø35
BBN-TENEX   LL-TX-2    AMES-TIP
```

For SITES it is possible to specify that only sites of a specific type (or types) are of interest.
The possible types are:

TENEX	PDP-1Ø TENEX systems
ITS	Incompatible Timesharing Systems
DEC1Ø	DEC PDP-1Ø systems
TIP	Terminal Interface Processors
MTIP	Magnetic tape TIPs

eg:

*SITES TENEX %
*%

THE FOLLOWING ARE UP:

BBN-TENEX BBN-TENEXB I4-TENEX USC-ISI

or:

*SITES TIP ITS %
*%

THE FOLLOWING ARE UP:

MIT-DMS AMES-TIP RADC-TIP ETAC-TIP GWC-TIP SAAC-TIP ARPA-TIP
MIT-AI MITRE-TIP NBS-TIP USC-TIP DOCB-TIP

HOSTS:

The HOSTS command is the same as the SITES command except that if no site types are specified then only sites which are not TIPs or MTIPs are typed.

DECIMAL.HOST.NUMBERS:

Causes decimal site address numbers to be typed with the site names for the SITES or HOSTS commands.

OCTAL.HOST.NUMBERS:

Causes octal site address numbers to be typed with the site names for the SITES or HOSTS commands.

NO.HOST.NUMBERS:

Causes no site address numbers to be typed with the site names for the SITES or HOSTS commands. This is the default case.

CONNECTIONS:

The connection information is given in the following tabular form:

@NETSTAT%

*CONNECTIONS
*%

ACTIVE CONNECTIONS:

I	STATE	LCL-SOCKET	HOST	4N-SOCKET	LNK	BITS-ALLOC	M-ALL	BS/VT
0	OPND	30324600305	BBN-TENEX	30324000142	5	944	6	B 8
5	LSNG	21						
6	LSNG	1						
13	OPND	30324000146	BBN-TENEX	30325000301	7	952	6	T106
20	OPND	30325000300	BBN-TENEX	30324000147	10	17728	2	B 8
21	OPND	30324000147	BBN-TENEX	30325000300	10	17696	2	T106
22	LSNG	3						
.								
.								
.								
111	OPND	30324000070	NBS-TIP	4200003	2	928	6	T101
116	OPND	30324000151	NBS-TIP	1600002	11	640	1	T102
117	OPND	30324000071	NBS-TIP	4200002	23	64	1	T101
121	LSNG	15						
130	OPND	33200101	BBN-TENEX	33200200	2	0	0	B36
134	RFN1	30324000153	SAAC-TIP	200002	3	8	0	B 8
135	LSNG	17						
136	OPND	33200200	BBN-TENEX	33200101	2	0	0	B36

where: I is the index of the connection in local system tables.

STATE is the state of the connection.

LCL-SOCKET is the absolute local socket number.

HOST is the name of the foreign host for this connection (or its address in octal if its name is not known to TENEX).

4N-SOCKET is the absolute foreign socket number.

LNK is the link of the connection.

BITS-ALLOC is the present number of bits (decimal) allocated for this connection.

M-ALL is the present number of messages allocated for this connection.

BS/VT is either the byte size for this connection or the virtual terminal number of the connection (assumed byte size of 8).

For LSNG connections only I, STATE, and LCL-SOCKET are given.

SOCKETS:

The SOCKETS command is the same as the CONNECTIONS command.

SPECIFIC CONNECTIONS:

It is possible to specify that only connections of specific types are of interest. This is done by typing:

```
*SPECIFIC CONNECTIONS:
**
```

NETSTAT will now accept specification commands.

The possible specifications are:

SITES or HOSTS to specify only connections with specific hosts or types of hosts. Names may be either the official network names or nicknames known to the local system. Host numbers are in octal.
eg:
**HOSTS UCLA-CCN 201 TENEX %

JOBS to specify only connections whose local socket numbers are relative to specific jobs. Job numbers are in decimal.
eg:

**JOBS 11,0%

SIZES to specify only connections which have given byte sizes. Byte sizes are in decimal.

eg:

**SIZES 8,32%

STATES to specify only connections which are in given states.
eg:

**STATES LSNG CLSW %

TTYS to specify only connections which are with specific virtual terminals. Terminal numbers are in octal.
eg:

**TTYS 101,105%

If no arguments are given for TTYS then all connections with virtual terminals are specified.
eg:

**TTYS %

USERS to specify only connections whose local socket numbers are relative to specific users.
eg:

**USERS JSMITH JDOE%

```
# Specification commands to select connections involving specific
# sockets are:
#
# LOCAL.SOCKETS                or
# SOCKETS            to select connections with specific local socket
#                 numbers
#
# FOREIGN.SOCKETS     to select connections with specific foreign socket
#                 numbers.
#
# The specification arguments for the SOCKETS commands can be any
# of the following (all numbers are octal):
#
# SKT             A single specific socket:  S = SKT
# SKT1-SKT2       Any socket in range:  SKT1 <= S <= SKT2
# SKT+INCR        Any socket in range:  SKT <= S <= SKT+INCR
# <SKT            Any socket in range:  S <= SKT
# >SKT            Any socket in range:  SKT <= S
```

The arguments for all of the above selection commands may be

separated by spaces, commas, or, for those arguments where altmode recognition may be done, by the altmode used in completing the previous argument. Altmode may be used to recognize all arguments which are not numbers, as well as the selection commands themselves. Only enough need be typed to uniquely identify the command or argument.

eg. The following are all equivalent and legal (ALTMODE is represented by "$"):

**ST$ATES LSNG,CLSW OPND%

**STATES L$SNG CLSW,O$PND %

**ST LS,CLS$W OP%

It is also possible to precede any of the above specification commands with the word "NOT". This has the effect of specifying all connections except those of the given type(s).

eg:

**NOT STATES LSNG OPND %

will specify all connections which are not in the LSNG or OPND states.

If a particular specification command has already been given without NOT, it is an error to later give the same command with NOT.

eg:

**HOSTS TENEX %
**NOT HOSTS ??
**

When more than one specification command is given, only the connections that satisfy all of the given specifications (ie. the logical AND of the conditions specified) are listed.

To terminate specifications and start typeout (or return to regular commands if no specifications were given) type carriage return on a blank line.

ie:
**%

REPEAT:

NETSTAT normally returns to EXEC when it completes its type out. This command causes NETSTAT to return to accept a new set of commands after each type out. Typing a blank line at that point will cause NETSTAT to again give the status information specified by the last set of commands.

QUIT:

Forces NETSTAT to return to EXEC.

BRIEF:

Causes no headings or messages to be typed when giving status information.

VERBOSE:

Causes headings to be always typed.
The default case is to print the heading for the connection information only the first time.

SPECIAL CHARACTERS:

^A (ctrl A) may be used to delete the last character typed.

? will prompt a help message.

RUBOUT will abort a line.

Besides causing recognition, ALTMODE will also prompt a description of the type of argument required when typed as the first character when an argument is expected.

^O (ctrl O) will stop the typeout. It acts independently on the site and the connection information.

: semi-colon can be used at the start of a line to indicate a line to be ignored.

Examples:

```
@NETSTAT%

*SPECIFIC CONNECTIONS:
**SIZES 8%
**NOT TTYS %
**%

ACTIVE CONNECTIONS:
  I STATE LCL-SOCKET     HOST       4N-SOCKET LNK  BITS-ALLOC M-ALL BS/VT

 15 OPND 30327000325 SRI-ARC     30324100024  27         184    19  B 8
 16 OPND 30327000305 BBN-TENEX   30324000140   5         952     6  B 8
 17 OPND 30327000324 SRI-ARC     30324100025   7       18168     2  B 8
 23 OPND 30327000304 BBN-TENEX   30324000141   6       18168     2  B 8
 25 OPND 30324000240 ETAC-TIP        400003   2           0     0  B 8
 33 OPND 30324000241 ETAC-TIP        400002   4           0     0  B 8
 45 OPND 30327000301 BBN-TENEX   30324000074   2         944     6  B 8
 50 OPND 30325500003 CMU-10B          400  62         392    19  B 8
 55 OPND 30327000300 BBN-TENEX   30324000075   3       18168     2  B 8
 57 OPND 30325500002 CMU-10B          401   3       17728     2  B 8
131 CLZW 30327000311 I4-TENEX    30324000056  36          48 262142  B 8
136 CLZW 30327000310 I4-TENEX    30324000057   3       18168     2  B 8


@NETSTAT%

*SPECIFIC CONNECTIONS:
**NOT HOSTS TIP %
**TTYS %
**%

ACTIVE CONNECTIONS:
  I STATE LCL-SOCKET     HOST       4N-SOCKET LNK  BITS-ALLOC M-ALL BS/VT

 13 OPND 30324000166 SRI-ARC     30326500003   2         952     6 T102
 21 OPND 30324000167 SRI-ARC     30326500002  17       12976    15 T102
 27 OPND 30324000220 SRI-ARC     30326500005   3         952     6 T103
 35 OPND 30324000221 SRI-ARC     30326500004  11       15832    17 T103
 64 OPND 30324000075 BBN-TENEX   30327000300   3       18168     2 T114
 65 CLZW 30324000315 CASE-10     30324600002  20       15736     6 T105
 66 OPND 30324000235 SRI-ARC     30326500010  53       14128    11 T101
111 OPND 30324000230 SRI-ARC     30326500007   4         904     6 T106
117 OPND 30324000231 SRI-ARC     30326500006   3       13752    20 T106
140 OPND 30324000074 BBN-TENEX   30327000301   2         944     6 T114
```

```
141 OPND 30324000314 CASE-10      30324600003   2          944      6 T105
142 OPND 30324000234 SRI-ARC      30326500011   5          944      6 T101


@NETSTAT%
*SPECIFIC CONNECTIONS %
**STATES OPND %
**SIZES 32%
**%
```

There are no connections of the specified type.

```
@NETSTAT%

*H TEN,TIP%
*S%

**JOBS 8
**


THE FOLLOWING  ARE UP:

SRI-ARC    BBN-TENEXB MITRE-TIP ETAC-TIP GWC-TIP  SAAC-TIP
UTAH-10    I4-TENEX   RADC-TIP  USC-ISI  DOCB-TIP ARPA-TIP
BBN-TENEX  AMES-TIP   NBS-TIP   USC-TIP

ACTIVE CONNECTIONS:
  I STATE LCL-SOCKET     HOST       4N-SOCKET LNK  BITS-ALLOC M-ALL BS/VT

 20 OPND 30325000300 BBN-TENEX   30324000375   4       17728      2  B 8
 45 OPND 30325000301 BBN-TENEX   30324000374   3         944      6  B 8


@NETSTAT%

*S%

**USERS TIP,BTHOMAS %
**%


ACTIVE CONNECTIONS:
  I STATE LCL-SOCKET     HOST       4N-SOCKET LNK  BITS-ALLOC M-ALL BS/VT

 23 LSNG    23100365
```

```
 32 LSNG      5600001
 46 LSNG      5600003
137 LSNG     23100371
```

```
@NETSTAT%

*SPECIFIC CONNECTIONS:
**NOT STATES LSNG %
**NOT HOSTS BBN%
**%
```

```
ACTIVE CONNECTIONS:
  I STATE LCL-SOCKET     HOST       4N-SOCKET LNK  BITS-ALLOC M-ALL BS/VT

 34 RFN1 30324000161 NBS-TIP         1600002  11         304      6 T102
 50 OPND 30325500003 USC-ISI    30324000020  31          96     20  B 8
 57 OPND 30325500002 USC-ISI    30324000021   3       17728      2  B 8
111 OPND 30324000070 NBS-TIP         4200003   2         720      6 T101
117 OPND 30324000071 NBS-TIP         4200002  23          48      0 T101
```

```
#                                RSEXEC
#
# RSEXEC is an experimental multi-computer Executive Program.  It
# contains several self-documenting features.  The following is a
# typescript of an RSEXEC session:
#
# @RSEXEC
#
#  RSEXEC  2.7.1  BBN-TENEX  TUE 25-JUN-74 09:03-EDT
#  Type HELP<cr> for help.
# _HELP
#
#  "?" gives a list of commands.
#
# Use the "DESCRIBE" command to obtain descriptions of other
# commands.  A good way to start is:
# _DESCRIBE RSEXEC<cr>
#
# Only enough of a command to uniquely identify it need be typed.
# "ESC"  invokes command recognition and completion.  Editing
# characters are:
#  ^A (Control A) - Character delete.
#  ^R (Control R) - Retypes current line or item.
# RUBOUT (or DEL) - Aborts current command (if typed while still
# giving command or arguments).
#
# ^C and ^T are handled by RSEXEC.
#
# ^P may be used as a panic escape in case your terminal becomes
# hung while linked.  It breaks the link, clears input and output
# buffers, and returns to the higher level EXEC.  The CONTINUE
# command will then resume the RSEXEC session as if a ^C had
# occurred.
# _?
# Commands are:
#  ACQUIRE
#  APPEND
#  BIND
#  BREAK
#  CONTINUE
#  COPY
#  DELETE
#  DESCRIBE
#  DEVSTAT
#  DIRECTORY
#  ENTER
#  ESCAPE
#  EXEC
#  EXPUNGE
#  FULLDUPLEX
#  GET
```