TELNET PROTOCOL SPECIFICATION

This RFC specifies a standard for the ARPA Internet community. Hosts on
the ARPA Internet are expected to adopt and implement this standard.

INTRODUCTION

   The purpose of the TELNET Protocol is to provide a fairly general,
   bi-directional, eight-bit byte oriented communications facility. Its
   primary goal is to allow a standard method of interfacing terminal
   devices and terminal-oriented processes to each other. It is
   envisioned that the protocol may also be used for terminal-terminal
   communication ("linking") and process-process communication
   (distributed computation).

GENERAL CONSIDERATIONS

   A TELNET connection is a Transmission Control Protocol (TCP)
   connection used to transmit data with interspersed TELNET control
   information.

   The TELNET Protocol is built upon three main ideas: first, the
   concept of a "Network Virtual Terminal"; second, the principle of
   negotiated options; and third, a symmetric view of terminals and
   processes.

   1. When a TELNET connection is first established, each end is
   assumed to originate and terminate at a "Network Virtual Terminal",
   or NVT. An NVT is an imaginary device which provides a standard,
   network-wide, intermediate representation of a canonical terminal.
   This eliminates the need for "server" and "user" hosts to keep
   information about the characteristics of each other's terminals and
   terminal handling conventions. All hosts, both user and server, map
   their local device characteristics and conventions so as to appear to
   be dealing with an NVT over the network, and each can assume a
   similar mapping by the other party. The NVT is intended to strike a
   balance between being overly restricted (not providing hosts a rich
   enough vocabulary for mapping into their local character sets), and
   being overly inclusive (penalizing users with modest terminals).

      NOTE: The "user" host is the host to which the physical terminal
      is normally attached, and the "server" host is the host which is
      normally providing some service. As an alternate point of view,

applicable even in terminal-to-terminal or process-to-process
communications, the "user" host is the host which initiated the
communication.

2. The principle of negotiated options takes cognizance of the fact
that many hosts will wish to provide additional services over and
above those available within an NVT, and many users will have
sophisticated terminals and would like to have elegant, rather than
minimal, services. Independent of, but structured within the TELNET
Protocol are various "options" that will be sanctioned and may be
used with the "DO, DON'T, WILL, WON'T" structure (discussed below) to
allow a user and server to agree to use a more elaborate (or perhaps
just different) set of conventions for their TELNET connection. Such
options could include changing the character set, the echo mode, etc.

The basic strategy for setting up the use of options is to have
either party (or both) initiate a request that some option take
effect. The other party may then either accept or reject the
request. If the request is accepted the option immediately takes
effect; if it is rejected the associated aspect of the connection
remains as specified for an NVT. Clearly, a party may always refuse
a request to enable, and must never refuse a request to disable some
option since all parties must be prepared to support the NVT.

The syntax of option negotiation has been set up so that if both
parties request an option simultaneously, each will see the other's
request as the positive acknowledgment of its own.

3. The symmetry of the negotiation syntax can potentially lead to
nonterminating acknowledgment loops -- each party seeing the incoming
commands not as acknowledgments but as new requests which must be
acknowledged. To prevent such loops, the following rules prevail:

   a. Parties may only request a change in option status; i.e., a
   party may not send out a "request" merely to announce what mode it
   is in.

   b. If a party receives what appears to be a request to enter some
   mode it is already in, the request should not be acknowledged.
   This non-response is essential to prevent endless loops in the
   negotiation. It is required that a response be sent to requests
   for a change of mode -- even if the mode is not changed.

   c. Whenever one party sends an option command to a second party,
   whether as a request or an acknowledgment, and use of the option
   will have any effect on the processing of the data being sent from
   the first party to the second, then the command must be inserted
   in the data stream at the point where it is desired that it take

effect. (It should be noted that some time will elapse between
the transmission of a request and the receipt of an
acknowledgment, which may be negative. Thus, a host may wish to
buffer data, after requesting an option, until it learns whether
the request is accepted or rejected, in order to hide the
"uncertainty period" from the user.)

Option requests are likely to flurry back and forth when a TELNET
connection is first established, as each party attempts to get the
best possible service from the other party. Beyond that, however,
options can be used to dynamically modify the characteristics of the
connection to suit changing local conditions. For example, the NVT,
as will be explained later, uses a transmission discipline well
suited to the many "line at a time" applications such as BASIC, but
poorly suited to the many "character at a time" applications such as
NLS. A server might elect to devote the extra processor overhead
required for a "character at a time" discipline when it was suitable
for the local process and would negotiate an appropriate option.
However, rather than then being permanently burdened with the extra
processing overhead, it could switch (i.e., negotiate) back to NVT
when the detailed control was no longer necessary.

It is possible for requests initiated by processes to stimulate a
nonterminating request loop if the process responds to a rejection by
merely re-requesting the option. To prevent such loops from
occurring, rejected requests should not be repeated until something
changes. Operationally, this can mean the process is running a
different program, or the user has given another command, or whatever
makes sense in the context of the given process and the given option.
A good rule of thumb is that a re-request should only occur as a
result of subsequent information from the other end of the connection
or when demanded by local human intervention.

Option designers should not feel constrained by the somewhat limited
syntax available for option negotiation. The intent of the simple
syntax is to make it easy to have options -- since it is
correspondingly easy to profess ignorance about them. If some
particular option requires a richer negotiation structure than
possible within "DO, DON'T, WILL, WON'T", the proper tack is to use
"DO, DON'T, WILL, WON'T" to establish that both parties understand
the option, and once this is accomplished a more exotic syntax can be
used freely. For example, a party might send a request to alter
(establish) line length. If it is accepted, then a different syntax
can be used for actually negotiating the line length -- such a
"sub-negotiation" might include fields for minimum allowable, maximum
allowable and desired line lengths. The important concept is that

such expanded negotiations should never begin until some prior
(standard) negotiation has established that both parties are capable
of parsing the expanded syntax.

In summary, WILL XXX is sent, by either party, to indicate that
party's desire (offer) to begin performing option XXX, DO XXX and
DON'T XXX being its positive and negative acknowledgments; similarly,
DO XXX is sent to indicate a desire (request) that the other party
(i.e., the recipient of the DO) begin performing option XXX, WILL XXX
and WON'T XXX being the positive and negative acknowledgments. Since
the NVT is what is left when no options are enabled, the DON'T and
WON'T responses are guaranteed to leave the connection in a state
which both ends can handle. Thus, all hosts may implement their
TELNET processes to be totally unaware of options that are not
supported, simply returning a rejection to (i.e., refusing) any
option request that cannot be understood.

As much as possible, the TELNET protocol has been made server-user
symmetrical so that it easily and naturally covers the user-user
(linking) and server-server (cooperating processes) cases. It is
hoped, but not absolutely required, that options will further this
intent. In any case, it is explicitly acknowledged that symmetry is
an operating principle rather than an ironclad rule.

A companion document, "TELNET Option Specifications," should be
consulted for information about the procedure for establishing new
options.

THE NETWORK VIRTUAL TERMINAL

   The Network Virtual Terminal (NVT) is a bi-directional character
   device. The NVT has a printer and a keyboard. The printer responds
   to incoming data and the keyboard produces outgoing data which is
   sent over the TELNET connection and, if "echoes" are desired, to the
   NVT's printer as well. "Echoes" will not be expected to traverse the
   network (although options exist to enable a "remote" echoing mode of
   operation, no host is required to implement this option). The code
   set is seven-bit USASCII in an eight-bit field, except as modified
   herein. Any code conversion and timing considerations are local
   problems and do not affect the NVT.

   TRANSMISSION OF DATA

      Although a TELNET connection through the network is intrinsically
      full duplex, the NVT is to be viewed as a half-duplex device
      operating in a line-buffered mode. That is, unless and until

options are negotiated to the contrary, the following default
conditions pertain to the transmission of data over the TELNET
connection:

1) Insofar as the availability of local buffer space permits,
data should be accumulated in the host where it is generated
until a complete line of data is ready for transmission, or
until some locally-defined explicit signal to transmit occurs.
This signal could be generated either by a process or by a
human user.

The motivation for this rule is the high cost, to some hosts,
of processing network input interrupts, coupled with the
default NVT specification that "echoes" do not traverse the
network. Thus, it is reasonable to buffer some amount of data
at its source. Many systems take some processing action at the
end of each input line (even line printers or card punches
frequently tend to work this way), so the transmission should
be triggered at the end of a line. On the other hand, a user
or process may sometimes find it necessary or desirable to
provide data which does not terminate at the end of a line;
therefore implementers are cautioned to provide methods of
locally signaling that all buffered data should be transmitted
immediately.

2) When a process has completed sending data to an NVT printer
and has no queued input from the NVT keyboard for further
processing (i.e., when a process at one end of a TELNET
connection cannot proceed without input from the other end),
the process must transmit the TELNET Go Ahead (GA) command.

This rule is not intended to require that the TELNET GA command
be sent from a terminal at the end of each line, since server
hosts do not normally require a special signal (in addition to
end-of-line or other locally-defined characters) in order to
commence processing. Rather, the TELNET GA is designed to help
a user's local host operate a physically half duplex terminal
which has a "lockable" keyboard such as the IBM 2741. A
description of this type of terminal may help to explain the
proper use of the GA command.

The terminal-computer connection is always under control of
either the user or the computer. Neither can unilaterally
seize control from the other; rather the controlling end must
relinquish its control explicitly. At the terminal end, the
hardware is constructed so as to relinquish control each time
that a "line" is terminated (i.e., when the "New Line" key is
typed by the user). When this occurs, the attached (local)

computer processes the input data, decides if output should be
generated, and if not returns control to the terminal. If
output should be generated, control is retained by the computer
until all output has been transmitted.

The difficulties of using this type of terminal through the
network should be obvious. The "local" computer is no longer
able to decide whether to retain control after seeing an
end-of-line signal or not; this decision can only be made by
the "remote" computer which is processing the data. Therefore,
the TELNET GA command provides a mechanism whereby the "remote"
(server) computer can signal the "local" (user) computer that
it is time to pass control to the user of the terminal. It
should be transmitted at those times, and only at those times,
when the user should be given control of the terminal. Note
that premature transmission of the GA command may result in the
blocking of output, since the user is likely to assume that the
transmitting system has paused, and therefore he will fail to
turn the line around manually.

The foregoing, of course, does not apply to the user-to-server
direction of communication. In this direction, GAs may be sent at
any time, but need not ever be sent. Also, if the TELNET
connection is being used for process-to-process communication, GAs
need not be sent in either direction. Finally, for
terminal-to-terminal communication, GAs may be required in
neither, one, or both directions. If a host plans to support
terminal-to-terminal communication it is suggested that the host
provide the user with a means of manually signaling that it is
time for a GA to be sent over the TELNET connection; this,
however, is not a requirement on the implementer of a TELNET
process.

Note that the symmetry of the TELNET model requires that there is
an NVT at each end of the TELNET connection, at least
conceptually.

STANDARD REPRESENTATION OF CONTROL FUNCTIONS

As stated in the Introduction to this document, the primary goal
of the TELNET protocol is the provision of a standard interfacing
of terminal devices and terminal-oriented processes through the
network. Early experiences with this type of interconnection have
shown that certain functions are implemented by most servers, but
that the methods of invoking these functions differ widely. For a
human user who interacts with several server systems, these
differences are highly frustrating. TELNET, therefore, defines a
standard representation for five of these functions, as described

below. These standard representations have standard, but not
required, meanings (with the exception that the Interrupt Process
(IP) function may be required by other protocols which use
TELNET); that is, a system which does not provide the function to
local users need not provide it to network users and may treat the
standard representation for the function as a No-operation. On
the other hand, a system which does provide the function to a
local user is obliged to provide the same function to a network
user who transmits the standard representation for the function.

Interrupt Process (IP)

   Many systems provide a function which suspends, interrupts,
   aborts, or terminates the operation of a user process. This
   function is frequently used when a user believes his process is
   in an unending loop, or when an unwanted process has been
   inadvertently activated. IP is the standard representation for
   invoking this function. It should be noted by implementers
   that IP may be required by other protocols which use TELNET,
   and therefore should be implemented if these other protocols
   are to be supported.

Abort Output (AO)

   Many systems provide a function which allows a process, which
   is generating output, to run to completion (or to reach the
   same stopping point it would reach if running to completion)
   but without sending the output to the user's terminal.
   Further, this function typically clears any output already
   produced but not yet actually printed (or displayed) on the
   user's terminal. AO is the standard representation for
   invoking this function. For example, some subsystem might
   normally accept a user's command, send a long text string to
   the user's terminal in response, and finally signal readiness
   to accept the next command by sending a "prompt" character
   (preceded by <CR><LF>) to the user's terminal. If the AO were
   received during the transmission of the text string, a
   reasonable implementation would be to suppress the remainder of
   the text string, but transmit the prompt character and the
   preceding <CR><LF>. (This is possibly in distinction to the
   action which might be taken if an IP were received; the IP
   might cause suppression of the text string and an exit from the
   subsystem.)

   It should be noted, by server systems which provide this
   function, that there may be buffers external to the system (in

the network and the user's local host) which should be cleared;
the appropriate way to do this is to transmit the "Synch"
signal (described below) to the user system.

Are You There (AYT)

Many systems provide a function which provides the user with
some visible (e.g., printable) evidence that the system is
still up and running. This function may be invoked by the user
when the system is unexpectedly "silent" for a long time,
because of the unanticipated (by the user) length of a
computation, an unusually heavy system load, etc. AYT is the
standard representation for invoking this function.

Erase Character (EC)

Many systems provide a function which deletes the last
preceding undeleted character or "print position"* from the
stream of data being supplied by the user. This function is
typically used to edit keyboard input when typing mistakes are
made. EC is the standard representation for invoking this
function.

    *NOTE: A "print position" may contain several characters
    which are the result of overstrikes, or of sequences such as
    <char1> BS <char2>...

Erase Line (EL)

Many systems provide a function which deletes all the data in
the current "line" of input. This function is typically used
to edit keyboard input. EL is the standard representation for
invoking this function.

THE TELNET "SYNCH" SIGNAL

Most time-sharing systems provide mechanisms which allow a
terminal user to regain control of a "runaway" process; the IP and
AO functions described above are examples of these mechanisms.
Such systems, when used locally, have access to all of the signals
supplied by the user, whether these are normal characters or
special "out of band" signals such as those supplied by the
teletype "BREAK" key or the IBM 2741 "ATTN" key. This is not
necessarily true when terminals are connected to the system
through the network; the network's flow control mechanisms may
cause such a signal to be buffered elsewhere, for example in the
user's host.

To counter this problem, the TELNET "Synch" mechanism is
introduced. A Synch signal consists of a TCP Urgent notification,
coupled with the TELNET command DATA MARK. The Urgent
notification, which is not subject to the flow control pertaining
to the TELNET connection, is used to invoke special handling of
the data stream by the process which receives it. In this mode,
the data stream is immediately scanned for "interesting" signals
as defined below, discarding intervening data. The TELNET command
DATA MARK (DM) is the synchronizing mark in the data stream which
indicates that any special signal has already occurred and the
recipient can return to normal processing of the data stream.

   The Synch is sent via the TCP send operation with the Urgent
   flag set and the DM as the last (or only) data octet.

When several Synchs are sent in rapid succession, the Urgent
notifications may be merged. It is not possible to count Urgents
since the number received will be less than or equal the number
sent. When in normal mode, a DM is a no operation; when in urgent
mode, it signals the end of the urgent processing.

   If TCP indicates the end of Urgent data before the DM is found,
   TELNET should continue the special handling of the data stream
   until the DM is found.

   If TCP indicates more Urgent data after the DM is found, it can
   only be because of a subsequent Synch. TELNET should continue
   the special handling of the data stream until another DM is
   found.

"Interesting" signals are defined to be: the TELNET standard
representations of IP, AO, and AYT (but not EC or EL); the local
analogs of these standard representations (if any); all other
TELNET commands; other site-defined signals which can be acted on
without delaying the scan of the data stream.

Since one effect of the SYNCH mechanism is the discarding of
essentially all characters (except TELNET commands) between the
sender of the Synch and its recipient, this mechanism is specified
as the standard way to clear the data path when that is desired.
For example, if a user at a terminal causes an AO to be
transmitted, the server which receives the AO (if it provides that
function at all) should return a Synch to the user.

Finally, just as the TCP Urgent notification is needed at the
TELNET level as an out-of-band signal, so other protocols which
make use of TELNET may require a TELNET command which can be
viewed as an out-of-band signal at a different level.

By convention the sequence [IP, Synch] is to be used as such a
signal. For example, suppose that some other protocol, which uses
TELNET, defines the character string STOP analogously to the
TELNET command AO. Imagine that a user of this protocol wishes a
server to process the STOP string, but the connection is blocked
because the server is processing other commands. The user should
instruct his system to:

     1. Send the TELNET IP character;

     2. Send the TELNET SYNC sequence, that is:

        Send the Data Mark (DM) as the only character
        in a TCP urgent mode send operation.

     3. Send the character string STOP; and

     4. Send the other protocol's analog of the TELNET DM, if any.

The user (or process acting on his behalf) must transmit the
TELNET SYNCH sequence of step 2 above to ensure that the TELNET IP
gets through to the server's TELNET interpreter.

     The Urgent should wake up the TELNET process; the IP should
     wake up the next higher level process.

   THE NVT PRINTER AND KEYBOARD

   The NVT printer has an unspecified carriage width and page length
   and can produce representations of all 95 USASCII graphics (codes
   32 through 126). Of the 33 USASCII control codes (0 through 31
   and 127), and the 128 uncovered codes (128 through 255), the
   following have specified meaning to the NVT printer:

NAME	CODE	MEANING
NULL (NUL)	0	No Operation
Line Feed (LF)	10	Moves the printer to the next print line, keeping the same horizontal position.
Carriage Return (CR)	13	Moves the printer to the left margin of the current line.

In addition, the following codes shall have defined, but not
required, effects on the NVT printer. Neither end of a TELNET
connection may assume that the other party will take, or will
have taken, any particular action upon receipt or transmission
of these:

BELL (BEL)   7   Produces an audible or
                                visible signal (which does
                                NOT move the print head).
Back Space (BS)   8   Moves the print head one
                                character position towards
                                the left margin.
Horizontal Tab (HT)   9   Moves the printer to the
                                next horizontal tab stop.
                                It remains unspecified how
                                either party determines or
                                establishes where such tab
                                stops are located.
Vertical Tab (VT)   11   Moves the printer to the
                                next vertical tab stop. It
                                remains unspecified how
                                either party determines or
                                establishes where such tab
                                stops are located.
Form Feed (FF)   12   Moves the printer to the top
                                of the next page, keeping
                                the same horizontal position.

All remaining codes do not cause the NVT printer to take any
action.

The sequence "CR LF", as defined, will cause the NVT to be
positioned at the left margin of the next print line (as would,
for example, the sequence "LF CR"). However, many systems and
terminals do not treat CR and LF independently, and will have to
go to some effort to simulate their effect. (For example, some
terminals do not have a CR independent of the LF, but on such
terminals it may be possible to simulate a CR by backspacing.)
Therefore, the sequence "CR LF" must be treated as a single "new
line" character and used whenever their combined action is
intended; the sequence "CR NUL" must be used where a carriage
return alone is actually desired; and the CR character must be
avoided in other contexts. This rule gives assurance to systems
which must decide whether to perform a "new line" function or a
multiple-backspace that the TELNET stream contains a character
following a CR that will allow a rational decision.

Note that "CR LF" or "CR NUL" is required in both directions

(in the default ASCII mode), to preserve the symmetry of the
NVT model. Even though it may be known in some situations
(e.g., with remote echo and suppress go ahead options in
effect) that characters are not being sent to an actual
printer, nonetheless, for the sake of consistency, the protocol
requires that a NUL be inserted following a CR not followed by
a LF in the data stream. The converse of this is that a NUL
received in the data stream after a CR (in the absence of
options negotiations which explicitly specify otherwise) should
be stripped out prior to applying the NVT to local character
set mapping.

The NVT keyboard has keys, or key combinations, or key sequences,
for generating all 128 USASCII codes. Note that although many
have no effect on the NVT printer, the NVT keyboard is capable of
generating them.

In addition to these codes, the NVT keyboard shall be capable of
generating the following additional codes which, except as noted,
have defined, but not required, meanings. The actual code
assignments for these "characters" are in the TELNET Command
section, because they are viewed as being, in some sense, generic
and should be available even when the data stream is interpreted
as being some other character set.

Synch

   This key allows the user to clear his data path to the other
   party. The activation of this key causes a DM (see command
   section) to be sent in the data stream and a TCP Urgent
   notification is associated with it. The pair DM-Urgent is to
   have required meaning as defined previously.

Break (BRK)

   This code is provided because it is a signal outside the
   USASCII set which is currently given local meaning within many
   systems. It is intended to indicate that the Break Key or the
   Attention Key was hit. Note, however, that this is intended to
   provide a 129th code for systems which require it, not as a
   synonym for the IP standard representation.

Interrupt Process (IP)

   Suspend, interrupt, abort or terminate the process to which the
   NVT is connected. Also, part of the out-of-band signal for
   other protocols which use TELNET.

Abort Output (AO)

Allow the current process to (appear to) run to completion, but
do not send its output to the user. Also, send a Synch to the
user.

Are You There (AYT)

Send back to the NVT some visible (i.e., printable) evidence
that the AYT was received.

Erase Character (EC)

The recipient should delete the last preceding undeleted
character or "print position" from the data stream.

Erase Line (EL)

The recipient should delete characters from the data stream
back to, but not including, the last "CR LF" sequence sent over
the TELNET connection.

The spirit of these "extra" keys, and also the printer format
effectors, is that they should represent a natural extension of
the mapping that already must be done from "NVT" into "local".
Just as the NVT data byte 68 (104 octal) should be mapped into
whatever the local code for "uppercase D" is, so the EC character
should be mapped into whatever the local "Erase Character"
function is. Further, just as the mapping for 124 (174 octal) is
somewhat arbitrary in an environment that has no "vertical bar"
character, the EL character may have a somewhat arbitrary mapping
(or none at all) if there is no local "Erase Line" facility.
Similarly for format effectors: if the terminal actually does
have a "Vertical Tab", then the mapping for VT is obvious, and
only when the terminal does not have a vertical tab should the
effect of VT be unpredictable.

TELNET COMMAND STRUCTURE

All TELNET commands consist of at least a two byte sequence: the
"Interpret as Command" (IAC) escape character followed by the code
for the command. The commands dealing with option negotiation are
three byte sequences, the third byte being the code for the option
referenced. This format was chosen so that as more comprehensive use
of the "data space" is made -- by negotiations from the basic NVT, of
course -- collisions of data bytes with reserved command values will
be minimized, all such collisions requiring the inconvenience, and

inefficiency, of "escaping" the data bytes into the stream. With the
current set-up, only the IAC need be doubled to be sent as data, and
the other 255 codes may be passed transparently.

The following are the defined TELNET commands. Note that these codes
and code sequences have the indicated meaning only when immediately
preceded by an IAC.

```
    NAME              CODE              MEANING

    SE                240     End of subnegotiation parameters.
    NOP               241     No operation.
    Data Mark         242     The data stream portion of a Synch.
                              This should always be accompanied
                              by a TCP Urgent notification.
    Break             243     NVT character BRK.
    Interrupt Process 244     The function IP.
    Abort output      245     The function AO.
    Are You There     246     The function AYT.
    Erase character   247     The function EC.
    Erase Line        248     The function EL.
    Go ahead          249     The GA signal.
    SB                250     Indicates that what follows is
                              subnegotiation of the indicated
                              option.
    WILL (option code) 251    Indicates the desire to begin
                              performing, or confirmation that
                              you are now performing, the
                              indicated option.
    WON'T (option code) 252   Indicates the refusal to perform,
                              or continue performing, the
                              indicated option.
    DO (option code)   253    Indicates the request that the
                              other party perform, or
                              confirmation that you are expecting
                              the other party to perform, the
                              indicated option.
    DON'T (option code) 254   Indicates the demand that the
                              other party stop performing,
                              or confirmation that you are no
                              longer expecting the other party
                              to perform, the indicated option.
    IAC               255     Data Byte 255.
```

CONNECTION ESTABLISHMENT

   The TELNET TCP connection is established between the user's port U
   and the server's port L. The server listens on its well known port L
   for such connections. Since a TCP connection is full duplex and
   identified by the pair of ports, the server can engage in many
   simultaneous connections involving its port L and different user
   ports U.

   Port Assignment

      When used for remote user access to service hosts (i.e., remote
      terminal access) this protocol is assigned server port 23
      (27 octal). That is L=23.

                     TELNET OPTION SPECIFICATIONS

This RFC specifies a standard for the ARPA Internet community. Hosts on
the ARPA Internet are expected to adopt and implement this standard.

The intent of providing for options in the TELNET Protocol is to permit
hosts to obtain more elegant solutions to the problems of communication
between dissimilar devices than is possible within the framework
provided by the Network Virtual Terminal (NVT). It should be possible
for hosts to invent, test, or discard options at will. Nevertheless, it
is envisioned that options which prove to be generally useful will
eventually be supported by many hosts; therefore it is desirable that
options should be carefully documented and well publicized. In
addition, it is necessary to insure that a single option code is not
used for several different options.

This document specifies a method of option code assignment and standards
for documentation of options. The individual responsible for assignment
of option codes may waive the requirement for complete documentation for
some cases of experimentation, but in general documentation will be
required prior to code assignment. Options will be publicized by
publishing their documentation as RFCs; inventors of options may, of
course, publicize them in other ways as well.

    Option codes will be assigned by:

        Jonathan B. Postel
        University of Southern California
        Information Sciences Institute (USC-ISI)
        4676 Admiralty Way
        Marina Del Rey, California 90291
        (213) 822-1511

        Mailbox = POSTEL@USC-ISIF

Documentation of options should contain at least the following sections:

    Section 1 - Command Name and Option Code

    Section 2 - Command Meanings

        The meaning of each possible TELNET command relevant to this
        option should be described. Note that for complex options, where

"subnegotiation" is required, there may be a larger number of
possible commands. The concept of "subnegotiation" is described
in more detail below.

Section 3 - Default Specification

The default assumptions for hosts which do not implement, or use,
the option must be described.

Section 4 - Motivation

A detailed explanation of the motivation for inventing a
particular option, or for choosing a particular form for the
option, is extremely helpful to those who are not faced (or don't
realize that they are faced) by the problem that the option is
designed to solve.

Section 5 - Description (or Implementation Rules)

Merely defining the command meanings and providing a statement of
motivation are not always sufficient to insure that two
implementations of an option will be able to communicate.
Therefore, a more complete description should be furnished in most
cases. This description might take the form of text, a sample
implementation, hints to implementers, etc.

A Note on "Subnegotiation"

Some options will require more information to be passed between hosts
than a single option code. For example, any option which requires a
parameter is such a case. The strategy to be used consists of two
steps: first, both parties agree to "discuss" the parameter(s) and,
second, the "discussion" takes place.

The first step, agreeing to discuss the parameters, takes place in
the normal manner; one party proposes use of the option by sending a
DO (or WILL) followed by the option code, and the other party accepts
by returning a WILL (or DO) followed by the option code. Once both
parties have agreed to use the option, subnegotiation takes place by
using the command SB, followed by the option code, followed by the
parameter(s), followed by the command SE. Each party is presumed to
be able to parse the parameter(s), since each has indicated that the
option is supported (via the initial exchange of WILL and DO). On
the other hand, the receiver may locate the end of a parameter string
by searching for the SE command (i.e., the string IAC SE), even if
the receiver is unable to parse the parameters. Of course, either
party may refuse to pursue further subnegotiation at any time by
sending a WON'T or DON'T to the other party.

Thus, for option "ABC", which requires subnegotiation, the formats of
the TELNET commands are:

    IAC WILL ABC

        Offer to use option ABC (or favorable acknowledgment of other
        party's request)

    IAC DO ABC

        Request for other party to use option ABC (or favorable
        acknowledgment of other party's offer)

    IAC SB ABC <parameters> IAC SE

        One step of subnegotiation, used by either party.

Designers of options requiring "subnegotiation" must take great care
to avoid unending loops in the subnegotiation process. For example,
if each party can accept any value of a parameter, and both parties
suggest parameters with different values, then one is likely to have
an infinite oscillation of "acknowledgments" (where each receiver
believes it is only acknowledging the new proposals of the other).
Finally, if parameters in an option "subnegotiation" include a byte
with a value of 255, it is necessary to double this byte in
accordance the general TELNET rules.