Network Working Group
Request for Comments: 1053

S. Levy
T. Jacobson
Minnesota Supercomputer Center
April 1988

Telnet X.3 PAD Option

Status of this Memo

This RFC proposes a new option to Telnet for the Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

1. Command name and code

X.3-PAD 30

2. Command meanings

IAC DO X.3-PAD

The issuing telnet requests that its peer perform X.3-PAD functions, or accepts an offer to do so.

IAC DON'T X.3-PAD

The issuing telnet demands that its peer not perform or cease performing X.3-PAD functions.

IAC WILL X.3-PAD

The issuing telnet offers to perform X.3-PAD functions or confirms that it will do so.

IAC WON'T X.3-PAD

The issuing telnet refuses to perform X.3-PAD functions or indicates that it is ceasing to handle them.

Typically a server (host) telnet will use DO and DON'T, while a client (user) telnet will use WILL and WON'T. For convenience, in the rest of this RFC 'host' and 'user' telnets refer to those saying 'DO X.3-PAD' or 'WILL X.3-PAD' respectively.

Both telnet peers may use this option without confusion, as all messages unambiguously identify whether they come from the host

("DO") or the user ("WILL") side.

Once DO and WILL have been exchanged, the host ("DO") telnet may send the following messages:

```
IAC SB  X.3-PAD  SET           <param1> <value1> ...  IAC SE
IAC SB  X.3-PAD  RESPONSE-SET  <param1> <value1> ...  IAC SE
IAC SB  X.3-PAD  SEND          IAC SE
```

while the user ("WILL") telnet may send the following messages:

```
IAC SB  X.3-PAD  IS            <param1> <value1> ...  IAC SE
IAC SB  X.3-PAD  RESPONSE-IS   <param1> <value1> ...  IAC SE
```

```
The code for SET          is 0
The code for RESPONSE-SET is 1
The code for IS           is 2
The code for RESPONSE-IS  is 3
The code for SEND         is 4
```

Messages listing parameter-value pairs may contain any number of such pairs, including zero. Each parameter and each value occupies one octet, except that 255 (IAC) is doubled wherever it appears.

3. Default conditions

The initial state is DON'T X.3-PAD, WON'T X.3-PAD. This RFC does not specify default values for most X.3 parameters. If the host telnet wishes a particular initial state (as it normally will), it should negotiate for it after exchange of DO/WILL messages.

X.3-PAD parameter values need not be preserved except when DO/WILL X.3-PAD is in effect. Thus if a host enables ("DO") X.3-PAD, negotiates about some parameters, then for some reason disables ("DONT") and later re-enables X.3-PAD, it must renegotiate any parameters it cares about.

Keeping in mind that the host telnet may not recognize all the parameters known to the user telnet, it is suggested that the user telnet's initial parameters allow a reasonable level of service even if they are never changed (e.g., it would be unwise to begin with all data forwarding conditions disabled). Extensions to X.3 should default to states resembling normal X.3 service where possible.

4. Motivation for the option

Where interactive terminals (or computers simulating them) are attached to host computers across a network, it is often desirable to provide them the same services as have long been provided for terminals directly attached to those hosts.

Many systems handle this by simply leaving all character processing to the host running the applications. Each character typed by the user is sent, often in its own packet, immediately to the host. This gives good control over interaction, but can cause a significant load on hosts and networks. Long-distance packet networks tend to be unreasonably slow or expensive or both when used in this mode.

Suitable character processing on the client (near the user's terminal) can greatly improve the situation. Unfortunately for standardization efforts, there are many possible approaches with differing purposes.

Some have already been proposed as Telnet options. The Remote Controlled Transmission and Echo option, [3], provides fine control over local buffering and echoing. The SUPDUP option, [4], offers a variety of input and display functions in terminal-independent form.

This RFC's proposal is intended to support efficient, approximate emulation, across a Telnet connection, of a host's normal handling of character-oriented terminals. Ideally, a user and an application program would not need to know whether they were linked by an RS-232 line or a TCP/IP network, except where the medium required a distinction (e.g., when establishing a connection).

Server implementors would wish for enough to emulate, purely locally, everything offered by their host's operating system; on the other hand, a standard calling on user telnets to provide all terminal handling functions of all known operating systems will find few implementors. One might settle on a subset of common operations, but which ones?

The CCITT world has used one approach to these problems: the set of PAD services defined by recommendation X.3. This RFC proposes that the Internet community adopt that solution to handle the same problems under Telnet. It is fairly simple, widely known and used, extensible, and solves most of the relevant problems.

Adopting X.3 would have another advantage. X.25 is the dominant worldwide standard interface between commercial packet networks and Internet systems, as evidenced by the DDN's adoption of X.25 basic and standard services as replacements for BBN 1822 and HDH. Telnet

and X.3 PAD traffic will have to coexist on X.25 networks; there will be a consequential desire for effective interoperation at the virtual terminal level. Extending Telnet along these lines would vastly simplify bridging the two.

Described here is a scheme for implementing X.3 services and negotiating their parameters across a Telnet connection.

5. Description of the option

Many, though not all, X.3 services are meaningful in the context of remote terminal processing; for some, it may be desirable to allow local control (by the user) but not remote control (by the server host). Some functions may not be provided, or provided in only limited form, by particular implementations. In general, an implementation should follow the Telnet norm of requesting desired service but continuing to function somehow in case it is not available.

Negotiations are asymmetrical. Since the user telnet is charged with local character handling while engaged in the session with the remote host, the X.3 parameters "belong" to the user side. The host telnet requests parameter changes with SET or RESPONSE-SET messages. Host requests might be on behalf of an application program, for example, disabling local echo while a password is being entered. The user telnet should give its "best effort" to accommodate these requests, but guarantees nothing except accurate status reporting.

A user telnet may allow the local user to request parameter changes too, though this RFC does not specify a way.

Where requests conflict, or where a user telnet cannot satisfy a request, the user telnet has the last word, since it does the relevant character processing. It may allow control from the host only, from the user only, may seek a compromise type of processing and so on, at the implementor’s discretion.

Host ("DO") telnets may also ask the user telnet to SEND its current parameter values. The user ("WILL") telnet must reply to each SEND message with a RESPONSE-IS message listing the values of all the parameters it knows about. It is strongly recommended that all parameters known to the telnet implementor be included in this list, even if their values cannot be changed. The intent is to give the host telnet the most complete information possible about the style of interaction which the user telnet is providing.

If possible, user telnets should also inform the server host (with an IS message) whenever local conditions (e.g., user requests) change a

parameter's state. This may not be feasible in some circumstances, and such behavior is negotiable -- see the discussion of parameter 0.

Note that there are no "error" messages defined in section 2. Almost all detectable errors (use of nonexisistent parameters or undefined values for known parameters) are indistinguishable from valid uses of options known to one peer but not the other. Hosts will normally wish to poll the user telnet's state after making a request anyway, so error responses do not seem to be needed.

The protocol messages listed in section 2 are to be used as follows.

SET and RESPONSE-SET ask the user telnet to change its values for the given X.3 parameters. The user telnet ignores unrecognized parameters; it sends no reply. The host sends SET to begin a negotiation, when some event on the host side causes a change in the desired set of parameters. The host sends RESPONSE-SET to continue negotiation, when it is dissatisfied with the user telnet's choice of parameters indicated in a RESPONSE-IS message. Typically, the host will test the user telnet's chosen behavior by issuing a SEND message following the SET or RESPONSE-SET, though the user telnet should not rely on this.

A SEND message from the host demands that the user telnet send RESPONSE-IS.

IS and RESPONSE-IS inform the host telnet of the current states of some or all of the user telnet's parameters. The user telnet sends IS when the user telnet changes a parameter for some local reason, e.g., at a request from the (human) user. An IS message may but need not list all parameters, e.g., it might list just those which changed.

It sends RESPONSE-IS in answer to each SEND request from the host. Every RESPONSE-IS should list ALL X.3-PAD parameters known to the user telnet implementor, even those which cannot be changed. Any host requests (SET or RESPONSE-SET) received before a SEND message should be processed before sending the answering RESPONSE-IS, so that their effects are reflected in the parameter values indicated there.

To permit synchronization (which SEND is this an answer to?), the user telnet should count SEND messages, and send exactly one RESPONSE-IS per SEND.

One might think that this protocol could be implemented with only SET, SEND and IS messages. The seemingly redundant RESPONSE-SET and RESPONSE-IS codes are needed to let both the user and host telnets distinguish new peer requests from attempts to renegotiate previous

actions, while preventing potential infinite negotiation loops.

SET and IS messages are sent when the host or user telnet wishes to inform its peer of a change in the X.3 processing mode desired by some higher level entity. This might happen at initialization, or on user or application-program request. The important thing is that these messages are NOT sent merely in response to another X.3-PAD message from the peer.

RESPONSE-SET and RESPONSE-IS messages should be sent in reply to a peer's [RESPONSE-]IS or SEND message. They reflect negotiation at the telnet level, rather than changes in the higher-level environment. A host which sends a SEND message may complain about the status indicated in the answering RESPONSE-IS by sending RESPONSE-SET but not SET.

Under this scheme, a possible rule for preventing infinite negotiations would be for the host to send at most zero, one, or some fixed number, of RESPONSE-SET messages following receipt of one IS message or one higher-level host-side request. After that, the host telnet simply accepts the user telnet's last offer as well as it can. Note that only the host needs to worry about loop prevention, since it does all the asking.

A given parameter should not be listed more than once in a single message.

A sample negotiation might look like this. (Here line breaks are not meaningful; ASCII carriage returns and line feeds are indicated by <CR> and <LF>; other characters stand for themselves. In the IAC SB octet values.)

```
        Host:   <CR><LF>%
                                          (User types "cd gibber<CR>")
        User:   cd gibber<CR><LF>
        Host:   Password required.<CR>LF>
                                             (Host disables echoing)
             IAC SB X.3-PAD SET 2 0 IAC SE
                                             (Host polls for status)
             IAC SB X.3-PAD SEND IAC SE
        User:                          (User telnet has disabled local
                                        echo.  Note that some
                                        parameters (e.g., 9, 10, 11)
                                        are not present, presumably
                                        unimplemented, and a few
                                        extension parameters
                                        (129, 134) in extension
                                        set 1 are also defined.)
              IAC SB X.3-PAD RESPONSE-IS 1 29 2 0 3 2 4 0 5 0 7 17 8 0
                                          12 0 13 3 15 1 16 8 17 21 18 0
                                          128 1 129 23 134 1 IAC SE
        Host:    password:
                                               (User types "squeak<CR>",
        User:    squeak<CR><LF>                    which is not echoed.)
        Host:                                  (Host re-enables echoing)
                 IAC SB X.3-PAD SET 2 1 IAC SE
                                               (Host polls for status)
                 IAC SB X.3-PAD SEND IAC SE
        User:
               IAC SB X.3-PAD RESPONSE-IS 1 29 2 1 3 2 4 0 5 0 7 17 8 0
                                           12 0 13 3 15 1 16 8 17 21 18 0
                                           128 1 129 23 134 1 IAC SE
```

6. Parameters

In outline, the X.3-PAD option uses the following parameters.

Parameter 0 indicates whether the user telnet notifies the host about parameter changes made for local reasons.

Parameters 1 through 22 are basically those of CCITT X.3, with some changes in interpretation; they are listed in detail below.

Parameters 23 through 127 are reserved for potential extensions to CCITT’s X.3 definition.

Parameter 128 selects an "extension set", determining the meaning of parameters 129 through 255. One extension set is proposed in this RFC, others may be added. The extension mechanism is explained under parameter 128’s description.

Parameters 129 through 255 are meaningful only when defined in the extension set indicated by the current value of parameter 128.

It should NOT be assumed that all implementations will necessarily support all parameters defined here, or all values of those parameters. Supported parameter/value combinations, however, should behave as described here.

The following parameter is specific to this Telnet option.

Parameter 0 -- Notify host of user-initiated parameter changes.

Code 0 -- Host is not notified.
Code 1 -- User telnet notifies host by sending IS message.

If the user telnet, for some local reason, changes a parameter, should it send an IS message to the host? This is desirable, since the host telnet cannot be sure of knowing the user telnet's current status otherwise. On the other hand, some user telnets may be unable to send notification. Consider a user calling from an X.25 PAD through an X.25-to-telnet gateway. The user may change local PAD parameters freely, but since normal PADs send no message when this happens, the gateway cannot inform the host telnet. Moreover, some sloppy host telnets may not wish to know about user parameter changes.

In normal usage, the host will ask to SET parameter 0 to its preferred state upon initialization; the user telnet accepts the setting if it can; then the host polls (using SEND) for the user telnet's decision. A disappointed host might periodically poll for changes, or admonish the (human) user not to change parameters, or remain silent and simply work oddly if changes are made.

The following parameters are as defined by the 1984 CCITT X.3 standard.

Numbers are in decimal.

Parameter 1 -- Character to escape to local telnet command mode.

Code 0 -- No ASCII character performs this function (though some special mechanism, e.g., a function key, still may).
Code 1 -- DLE (ASCII code 16).
Codes 32 through 126 -- ASCII code of the character.

Codes 2 through 31 are not defined by X.3, but might also be taken to refer to the corresponding ASCII control characters. X.3 seems to be unable to name SOH (control-A) as a command escape character.

Parameter 2 -- Local echo of characters typed by the user.

Code 0 -- No local echo.
Code 1 -- Local echo.

Several echoing styles are possible. Parameter 13 selects whether a carriage return echoes as itself or as CR LF. Parameter 20 may suppress echoing of particular ASCII characters. The extension parameter 134 selects a style for echoing non-printing characters such as ESC.

Parameter 3 -- Set of forwarding characters.

The value is bit-encoded; each nonzero bit specifies a set of characters. The user telnet should accept characters from the user's keyboard, buffering them until it receives any of the specified characters (or until some other forwarding condition is satisfied, see below), and then sending the buffer to the host.

It may forward earlier if necessary, e.g., if it runs out of buffer space. It MUST eventually forward after receiving one of the indicated characters.

Code 0 -- No forwarding characters.
Code 1 -- Alphanumeric characters (a-z, A-Z, 0-9).
Code 2 -- CR.
Code 4 -- ESC, BEL, ENQ, ACK.
Code 8 -- DEL, CAN, DC2.
Code 16 -- ETX, EOT.
Code 32 -- HT, LF, VT, FF.
Code 64 -- ASCII character codes 0 through 31 not listed above.

Note that there is no way provided here to forward on printable, non-alphanumeric characters (punctuation marks).

Codes may be added to select the union of the associated sets of characters.

Parameter 4 -- Forward after idle time.

When this parameter is nonzero, the user telnet sends its input buffer to the host after a given period in which no characters are typed, even if no forwarding character was received.

Code 0 -- Infinite time limit.
Codes 1 through 255 -- Time limit in 1/20 second units.

The value "1" may be taken to mean "forward immediately" if timed

input is inconvenient to provide. For other values, when timing is available but the exact requested value is not, rounding to larger time delays is suggested. If timing is requested but none is available, immediate forwarding on every character is much preferred over an infinite time limit.

Note the interaction with parameter 15, Local editing, and the notes made under that heading.

Parameter 5 -- Flow control of user-to-host data.

A user telnet may be overwhelmed by data typed by the user. If parameter 5 is 1, it may output X-OFF (DC3, ASCII code 19) to ask the user to suspend input and X-ON (DC1, ASCII code 17) when the user may resume typing.

Code 0 -- X-OFF and X-ON considered normal output data.
Code 1 -- X-OFF and X-ON used to control user input.

The extension parameters 130 and 131, if defined, specify other codes to be used instead of ASCII DC3 and DC1.

Parameter 6, referring to messages sent from the PAD to the user, does not seem to be relevant in this context.

Parameter 7 -- Function of Break, Interrupt, Attention, etc.

This parameter describes handling of some special key or other character, implementation-defined, on the user's keyboard. It is bit-encoded; codes may be added to select multiple functions. Multiple functions may be performed in any order. Any messages generated should be promptly sent to the host.

Code 0 -- No action.
Code 1 -- Send interrupt packet (Telnet IAC IP).
Code 2 -- Reset (break Telnet connection).
Code 4 -- Discard input from user not yet consumed by host.
Code 8 -- Escape to local Telnet command mode.
Code 16 -- Discard output from host (see parameter 8).

The X.25 'Interrupt', 'Reset', and 'Indication of Break' messages are here translated to Telnet equivalents. See section 8 for suggestions on discarding input and output.

Parameter 8 -- Discarding output from host.

This parameter is intended as a flag, indicating whether host output is being ignored.

Code 0 -- Host output is sent to user.
Code 1 -- Host output is discarded.

This parameter is normally used in conjunction with parameter 7 when the 16's bit (Discard output on Break/Interrupt/Attention) is set. An implementation is suggested in section 8 of this RFC.

Note that, if a signal from the user causes parameter 8 to be changed and parameter 0 is set to 1, an X.3-PAD IS message should be sent to the host.

Parameter 9 -- Padding after carriage return.

This parameter selects insertion of ASCII NUL padding characters after output of each carriage return.

Codes 0 through 7 -- Insert that many padding characters.

Parameter 10 -- Line folding.

Output lines may be folded (e.g., by insertion of carriage return and line feed) when they exceed a specified width.

Code 0 -- No output line folding.
Codes 1 through 255 -- Fold lines after that many characters.

Parameter 11 -- Bit rate.

This parameter indicates the serial data rate of the user's terminal, if any. Though CCITT X.3 considers this parameter to be read-only, it may be meaningful to allow the host to set as well as read this value, thus changing the user's line speed dynamically.

Code 0 -- 110 bps	Code 10 -- 50 bps
Code 1 -- 134.5 bps	Code 11 -- 75 bps in, 1200 out
Code 2 -- 300 bps	Code 12 -- 2400 bps
Code 3 -- 1200 bps	Code 13 -- 4800 bps
Code 4 -- 600 bps	Code 14 -- 9600 bps
Code 5 -- 75 bps	Code 15 -- 19200 bps
Code 6 -- 150 bps	Code 16 -- 48000 bps
Code 7 -- 1800 bps	Code 17 -- 56000 bps
Code 8 -- 200 bps	Code 18 -- 64000 bps
Code 9 -- 100 bps	

Parameter 12 -- Flow control of host-to-user data.

When this parameter is 1, the user may type X-OFF (DC3, ASCII code 19) to suspend printing output, and X-ON (DC1, ASCII code 17) to resume output.

Code 0 -- X-OFF and X-ON are sent as data to host.
Code 1 -- X-OFF and X-ON control output.

See also the extension parameters 130, 131 and 132.

Parameter 13 -- Line feed insertion; Telnet CR LF vs CR NUL.

The CCITT uses this parameter to select whether a typed CR should be sent as CR or CR-LF, whether an output CR should have a LF printed after it, and whether an echoed CR should be echoed with an accompanying LF.

Here, it resolves the questions of mapping between the Telnet CR-LF sequence and single ASCII codes (i.e., when the user presses the carriage return key, should CR LF or CR NUL be sent to the host? When the host sends CR LF, should the user see CR LF or merely CR?)

The value is bit-encoded; codes may be added to select multiple functions.

Code 0 -- No line feed insertion
(typed CR sent as CR NUL; host CR LF printed as CR).
Code 1 -- Add line feed on output (host CR LF printed as CR LF).
Code 2 -- Add line feed on input (typed CR sent as CR LF to host).
Code 4 -- When echoing a typed CR locally, echo as CR LF.

Note the interaction with the TRANSMIT-BINARY Telnet option [5]. If the host has said WILL TRANSMIT-BINARY, then CR has no special meaning on output; it always stands for the single character CR regardless of this parameter's value. If the user telnet has said WILL TRANSMIT-BINARY, a typed CR should likewise always be sent as itself and not as CR LF or CR NUL.

Parameter 14 -- Output padding after line feed.

Gives the number of ASCII NUL padding characters to be sent to the user's terminal after each output line feed.

Codes 0 through 7 -- Send that many padding characters.

Parameter 15 -- Local editing.

If this parameter is 1, the character delete, line delete and line reprint functions (parameters 16, 17 and 18), if implemented, should be enabled. Data should be sent to the host when a forwarding character (see parameter 3) is typed or in case the user telnet's input buffer becomes full.

Note the interaction with parameter 4, Forward after idle time. User telnets need not handle the case where idle-time forwarding and local editing are both enabled, i.e., the host should explicitly request changing parameter 4 to 0 along with setting parameter 15 to 1.

Code 0 -- No input editing. Any editing characters are considered data.
Code 1 -- Input editing. Editing characters edit the input buffer.

Parameter 16 -- Character-delete character.

While local editing (parameter 15) is enabled, typing this character erases the last character in the editing buffer, if any. When editing is disabled, this character is not treated specially.

Code 0 -- No character has this function.
Codes 1 through 127 -- ASCII code of character-delete character.

See also parameter 19.

Parameter 17 -- Line-delete character.

While local editing (parameter 15) is enabled, this character erases the entire contents of the editing buffer. When editing is disabled, this character is not treated specially.

Code 0 -- No character has this function.
Codes 1 through 127 -- ASCII code of line-delete character.

See also parameter 19.

Parameter 18 -- Line-display character.

While local editing (parameter 15) is enabled, typing this character causes the current contents of the editing buffer to be printed on the user's terminal; nothing is sent to the host. When editing is disabled, this character is not treated specially.

Code 0 -- No character has this function.

Codes 1 through 127 -- ASCII code of line-display character.

Parameter 19 -- Editing service signals.

This determines what is echoed to the user when local editing is enabled and the character-delete or line-delete character is entered.

Code 0 -- Nothing is echoed.
Code 1 -- Editing style is suitable for printing terminals.
Code 2 -- Editing style is suitable for display terminals.
Codes 8 and 32-126 -- Echo that ASCII character for character-delete.

X.3 is specific on handling character- and line-deletion. If parameter 19 is 1, echo character-delete with a "line delete with three X's followed by CR LF. If 2, a character-delete echoes BS SPACE BS, while a line delete echoes enough BS SPACE BS's to erase the entire line. If 8 or 32-126, character-delete echoes that character, and line delete echoes XXX CR LF. An extension parameter could override these, selecting other styles if desired, though none is proposed here.

Parameter 20 -- Echo mask.

When local echoing, parameter 2, is enabled, each nonzero bit in this bit-encoded parameter's value suppresses echoing of some subset of ASCII characters. Adding values suppresses echo for the union of the specified subsets.

Code 0 -- all ASCII characters are echoed.
Code 1 -- CR is not echoed.
Code 2 -- LF is not echoed.
Code 4 -- VT, HT, and FF are not echoed.
Code 8 -- BEL and BS are not echoed.
Code 16 -- ESC and ENQ are not echoed.
Code 32 -- ACK, NAK, STX, SOH, EOT, ETB and ETX are not echoed.
Code 64 -- Editing characters are not echoed.
Code 128 -- other non-printing ASCII characters, and DEL, not echoed.

Nothing is echoed when parameter 2 is 0. Some characters should not be echoed regardless of parameter 20. If any of parameters 5, 12, or 22 are enabled (non-zero), then the XON and XOFF characters should not be echoed. Nor should the escape-to-local command mode character, parameter 1.

Parameter 21 -- Parity.

This parameter determines whether parity is checked on user input and generated on output to the user. Values may be added to select both.

Code 0 -- Parity neither generated nor checked.
Code 1 -- Even parity checked on input.
Code 2 -- Even parity generated on output.

Parameter 22 -- Page wait.

If enabled, this parameter causes the user telnet to pause after every N lines of output as if X-OFF had been received. Output resumes when X-ON is typed.

Code 0 -- No pause.
Codes 1-255 -- Pause after output of that many line feeds.

See also parameters 130, 131 and 132.

The following parameters are not part of CCITT X.3, but use the extension mechanism proposed for this Telnet option.

Parameter 128 -- Extension set number.

This parameter selects one of a potentially large number of "extension sets" -- more or less coherent collections of parameters added to the basic X.3 family. User telnets may support several extension sets. The host may determine whether a particular one is supported by trying to set parameter 128. The user telnet should accept the value if it provides some or all of the parameters in that set.

Extension sets might be defined for a variety of purposes. For example, Berkeley UNIX tty emulation, VMS emulation, Telenet's extended parameters, French national PDN parameters, and so on.

Initial values need not be specified for extension parameters (i.e., a host should explicitly negotiate for their values after selecting an extension set). However, it is recommended that default settings give service that resembles normal CCITT X.3 behavior where possible.

Extension sets are mutually exclusive. Different sets may use the same parameters (from 129 through 255) for different purposes.

Only one extension set is in effect at a time. That is, if a host

requests service X from extension set A, then switches to extension set B and requests its service Y, it should not expect that service X is still being provided.

Some values of this parameter are reserved:

Code 0 -- Null extension set. Only (a subset of) the basic CCITT X.3 parameters is provided. Every user telnet should accept this setting.
Code 1 -- (A subset of) the extension set 1 parameters described below is provided.
Code 255 -- Reserved for purely local (e.g., to a site), non-standard collections of extensions.

Other extension sets may be proposed and assigned set numbers in the range 2 through 254.

Set number are registered with the Internet Assigned Numbers Coordinator at USC-ISI. Please contact:

Joyce K. Reynolds
USC Information Sciences Institute
Suite 1001
4676 Admiralty Way
Marina del Rey, CA 90292-6695

213-822-1511 JKReynolds@ISI.EDU

The following parameters form extension set number 1.

Parameter 129, extension set 1 -- Word-delete character.

Typing this character while local editing is enabled causes the last word in the editing buffer to be erased. Several definitions for a "word" are in common use; this RFC does not specify one. There should be an indication to the user of what was erased. When editing is disabled, this character is not treated specially.

Code 0 -- No character has this function.
Codes 1 through 127 -- ASCII code of word-delete character.

Parameter 130, extension set 1 -- Flow control OFF character.

Parameter 131, extension set 1 -- Flow control ON character.
Typing these characters while parameter 12 is enabled cause output to be suspended or resumed, respectively. The user telnet may send them to the user while parameter 5 is enabled to ask the user to cease or resume supplying input.

If defined, these parameters should have default values of 19 (ASCII DC3) for parameter 130, and 17 (ASCII DC1) for parameter 131.

Code 0 -- No character has this function.
Codes 1 through 127 -- Function performed by that ASCII code.

Parameter 132, extension set 1 -- Host-to-user flow control convention.

Some styles of flow control accept only a particular character (e.g., X-ON) to resume output; others resume on receipt of any character. This parameter selects which to use. The default should be zero, as this matches the X.3 convention.

Code 0 -- Resume output only when correct character is received.
Code 1 -- Resume output when any character is received.

Parameter 133, extension set 1 -- Alternate Attention, etc., character.

This character serves as a synonym for the Break, Attention, etc., key whose function is given by parameter 7.

Code 0 -- No ASCII character has this function
(there may still be a special key or other mechanism).
Codes 1 through 127 -- ASCII code of the character.

Parameter 134, extension set 1 -- Local echo style.

This parameter selects how non-printing characters should be echoed, when parameter 2 is set to 1. The default should be zero, where all characters are simply echoed as themselves (except possibly carriage return; see parameter 13).

Code 0 -- All characters echo as themselves.
Code 1 -- Non-editing control characters echo as ^X for some printable character X.

See also parameters 2, Local echo, and 20, Echo mask, which may suppress echo of some or all characters regardless of this parameter.

Parameter 135, extension set 1 -- Accept following character as data.

After typing this character, the next character entered is accepted as data for transmission to the host even if it would normally have a special meaning.

The default should be zero.

Code 0 -- No character has this function.
Codes 1 through 127 -- ASCII code of the character.

Parameter 136, extension set 1 -- Character to toggle discarding output.

Typing this character changes the state of parameter 8 (discarding host-to-user output) from 0 to 1 or from 1 to 0. Thus an indeterminate amount of host output, received between successive instances of this character, will be discarded.

As usual, the host should be notified of each change if parameter 0 is set to 1. The host might wish to send SET messages at appropriate points (e.g., preceding command prompts) to re-enable output.

The default should be zero.

Code 0 -- No character performs this function (though another mechanism still may do so).

Codes 1 through 127 -- Typing that character toggles parameter 8.

Parameter 137, extension set 1 -- User-to-host bits per character.

Parameter 138, extension set 1 -- Host-to-user bits per character.

These parameters determine whether, for example, a full 8-bit input or output data path is available, or whether the most significant bit(s) of input or output data is stripped. Typical values would be 7 or 8.

Note that an 8-bit data path does not by itself imply transparent input/output; CR -> CR LF translation, XON/XOFF interpretation, parity and so on must also be disabled to achieve this.

7. Subsets, Extensions and Conflicts

An option as complex (and easy to extend) as this one, needs a policy for what subsets and extensions are allowed, and recommendations for negotiating one's way through a maze of partial implementations. In short, what does it mean to say DO or WILL X.3-PAD?

A basic principle is that, since hardly any user telnet implementation will provide all possible features, a host cannot expect to get precisely any desired kind of service.

[This may be an arguable point. The CCITT defines a mandatory subset of supported values for each X.3 parameter, with further

values optional. For example, the set of forwarding characters, parameter 4, must accept at least the values 0 (none), 2 (carriage return), and 126 (any control character or DEL). Though it would be possible to adopt the CCITT's set of mandatory values there, I don't think that would be desirable for two reasons.

First, some of the features specified (e.g., timed input) may be hard to implement in some environments, and may well not be necessary for many applications.

Second, this option provides for definition of entirely new parameters. Unlike the X.3 case, one peer may use parameters whose very existence is unknown to the other. So one cannot specify mandatory or default values for ALL parameters.]

On the other hand, a host is at least entitled to know what kind of service is being provided to the ultimate user. A user telnet's status report may be incomplete (not describing features its implementor did not know of); it may not describe the style of interaction the host (or user, or application) would wish for, but it should at least describe reality.

For telnet parameters with a range of possible values, if a user telnet implements only one "enabled" and one "disabled" value, it should choose the "enabled" value when asked for a setting it cannot supply. A VMS telnet, for example, might allow only DEL or nothing as the character-delete code. If a host asks it to use "backspace", it should choose DEL rather than nothing. The host may then interpret this contrary behavior as indicating a preferred value.

The problem of conflicting parameters, where several parameters control overlapping services and may conflict, is a serious one. The extension set scheme (see parameter 128) is intended to limit the problem. Each extension set's parameters should be selfconsistent and consistent with the CCITT X.3 parameters, but separate extension sets need not be concerned with each other's parameters.

Where parameters might conflict, it is important to specify priority as part of the parameters' description. For example, among parameters 2 (Local echo), 20 (Echo mask), and extension set 1's 134 (Local echo style), Echo mask is significant only if Local echo is enabled, and Local echo style applies only to characters selected for echoing by the first two parameters.

This option's functions overlap with those of some existing Telnet options, for example, ECHO (which can be interpreted to affect local echo and possibly local line editing), NAOCRD and NAOLFD [6] (specifying padding after output carriage returns and line feeds),

TRANSMIT-BINARY, Remote Controlled Transmission and Echo [3], and SUPDUP [4].

Where X.3-PAD completely subsumes the function of another option, as for ECHO, NAOCRD and NAOLFD, it's probably best to let the X.3-PAD option, where acceptable to both sides, supplant them and to refuse the other option.

The TRANSMIT-BINARY option can change (actually suppress) the interpretation of some bits of parameter 13 related to Telnet newline encoding, as mentioned under that parameter. As such it is compatible with this option but must be kept in mind.

RCTE would be a much more difficult case, since its service does not fit into this option's scheme and vice versa. However, it probably is unimportant because of the scarcity of RCTE implementations.

Some existing Telnet options can serve related but complementary functions, for example NAOHTS [7] for output tab handling, or TERMINAL-TYPE [8].

8. Implementation suggestions

It is strongly recommended that a user telnet support at least the combination with parameters 2=0, 3=126, and 4=1 (no local echo, forward immediately or nearly so on any input character) so that a dissatisfied host has the option of backing off and doing its own character handling.

The "discard output" function invoked by the Break, Interrupt, Attention, etc., key if the 16's bit is set in parameter 7 may be implemented as follows.

1. When the key is pressed, set parameter 8 to 1, begin discarding output, send IAC SB X.3-PAD IS 8 1 IAC SE to notify the host. (It may not need to know, but the message should be sent for consistency.)

2. Send IAC DO TIMING-MARK.

3. Send any other messages associated with the key (e.g., IAC IP).

4. Eventually, the host should send either IAC WILL TIMING-MARK or IAC WON'T TIMING-MARK, even if it knows nothing about the TIMING-MARK option. It will probably appear close, in the output stream, to the point where the host recognized any associated messages (e.g., IP).

When the TIMING-MARK arrives, reset parameter 8 to 0 and cease discarding output. Send IAC SB X.3-PAD IS 8 0 IAC SE.

The Telnet SYNCH mechanism (see [2]) may be employed in concert with such a scheme. A closed-loop flush, though, will be more effective at discarding excess output than SYNCH used alone. Provision of some such mechanism for discarding unwanted output, e.g., after interrupting the host, is heartily recommended.

Discarding input is less clear cut. Certainly, any buffered data not yet sent should be discarded; one might also use SYNCH to encourage the host telnet to discard more.

9. References

1. Recommendation X.3, from International Telecommunications Union CCITT Red Book, volume VIII, fascicle VIII.2, 1984.

2. Postel, J., and J. Reynolds, "Telnet Protocol Specification", RFC-854, USC Information Sciences Institute, May 1983.

3. Postel, J., and D. Crocker, "Remote Controlled Transmission and Echoing Telnet Option", RFC-726 and NIC-39237, SRI-ARC, March 1977.

4. Crispin, M., "SUPDUP Protocol", RFC-734 and NIC-41953, SU-AI October 1977; Crispin, M., "Telnet SUPDUP Option", RFC-736 and NIC-42213, SU-AI, October 1977; also Greenberg, B., "Telnet SUPDUP-OUTPUT Option", RFC-749 and NIC-45499, MIT-Multics, September 1978.

5. Postel, J., and J. Reynolds, "Telnet Binary Transmission Option", RFC-856, USC Information Sciences Institute, May 1983.

6. Crocker, D., "Telnet Output Linefeed Disposition Option", RFC-658 and NIC-31161, UCLA-NMC, October 1974; and "Telnet Output Carriage Return Disposition Option", RFC-652 and NIC-31155, UCLA-NMC, October 1974.

7. Crocker, D., "Telnet Output Horizontal Tab Stops Option", RFC-653 and NIC-31156, UCLA-NMC, October 1974. [RFC numbers 652 through 658 (NIC 31155 through 31161) are in a similar vein.]

8. Solomon, M., and E. Wimmers, "Telnet Terminal Type Option", RFC-884, University of Wisconsin - Madison, December 1983.