TN3270E Working Group
Internet Draft: <draft-ietf-tn3270e-tn5250e-01.txt>
Updates:  RFC 1205
Expiration Date: August, 1998

T. Murphy, Jr.
P. Rieth
J. Stevens
IBM Corporation

February 1998

# 5250 Telnet Enhancements

## Status of this Memo

## Copyright Notice

## Abstract

This draft describes the interface to the IBM 5250 Telnet server that allows client Telnet to request a Telnet terminal or printer session using a specific device name.  If a requested device name is not available, a method to retry the request using a new device name is described.  Methods to request specific Telnet session settings and auto-signon function are also described.

By allowing a Telnet client to select the device name, the 5250 Telnet server opens the door for applications to set and/or extract useful information about the Telnet client.  Some possibilities are 1) selecting a customized device name associated with a particular user profile name for National Language Support or subsystem routing, 2) connecting PC and network printers as clients and 3) auto-signon using clear-text or DES-encrypted password exchange.

Applications may need to use system API's on the AS/400 in order to extract Telnet session settings from the device name description. Refer to the Retrieve Device Description (QDCRDEVD) API described in the AS/400 System API book [3] on how to extract information using the DEVD0600 and DEVD1100 templates.

This draft describes how the IBM 5250 Telnet server supports Work Station Function (WSF) printers using 5250 Display Station Pass-Through.  A response code is returned by the Telnet server to indicate success or failure of the WSF printer session.

## 1. Table of Contents

CONTENTS

LIST OF FIGURES

## 2. Enhancing Telnet Negotiations

The 5250 Telnet server enables clients to negotiate both terminal and printer device names through Telnet Environment Options Negotiations, defined in the Standards Track RFC 1572 [13].

The purpose of RFC 1572 is to exchange environment information using a set of standard or custom variables. By using a combination of both standard VAR's and custom USERVAR's, the 5250 Telnet server allows client Telnet to request a pre-defined specific device by name.

If no pre-defined device exists then the device will be created, with client Telnet having the option to negotiate device attributes, such as the code page, character set, keyboard type, etc.

Since printers can now be negotiated as a device name, new terminal types have been defined to request printers. For example, you can now negotiate "IBM-3812-1" and "IBM-5553-B01" as valid TERMINAL-TYPE options [11].

Finally, the 5250 Telnet server will allow exchange of user profile and password information, where the password may be in either clear-text or encrypted form. If a valid combination of profile and password is received, then the client is allowed to bypass the sign-on panel. The setting of the QRMTSIGN system value must be either *VERIFY or *SAMEPRF for the bypass of the sign-on panel to succeed.

## 3. Standard Telnet Option Negotiation

Telnet server option negotiation typically begins with the issuance, by the server, of an invitation to engage in terminal type negotiation with the Telnet client (DO TERMINAL-TYPE) [11]. The client and server then enter into a series of sub-negotiations to determine the level of terminal support that will be used. After the terminal type is agreed upon, the client and server will normally negotiate a required set of additional options (EOR [12], BINARY [10], SGA [15]) that are required to support "transparent mode" or full screen 5250/3270 block mode support. As soon as the required options have been negotiated, the server will suspend further negotiations, and begin with initializing the actual virtual device on the AS/400. A typical exchange might start like the following:

```
AS/400 Telnet server             Enhanced Telnet client
--------------------------       -------------------------
IAC DO TERMINAL-TYPE        -->
                            <--  IAC WILL TERMINAL-TYPE
IAC SB TERMINAL-TYPE SEND
IAC SE                      -->
                                 IAC SB TERMINAL-TYPE IS
                            <--  IBM-5555-C01 IAC SE
IAC DO EOR                  -->
                            <--  IAC WILL EOR
                            <--  IAC DO EOR
IAC WILL EOR                -->
                             .
                             .
(other negotiations)         .
```

Actual bytes transmitted in the above example are shown in hex below.

```
AS/400 Telnet server             Enhanced Telnet client
--------------------------       -------------------------
FF FD 18                    -->
                            <--  FF FB 18
FF FA 18 01 FF F0           -->
                                 FF FA 18 00 49 42 4D 2D
                                 35 35 35 35 2D 43 30 31
                            <--  FF F0
FF FD 19                    -->
                            <--  FF FB 19
                            <--  FF FD 19
FF FB 19                    -->
                             .
                             .
(other negotiations)         .
```

Some negotiations are symmetrical between client and server and some are negotiated in one direction only.  Also, it is permissible and common practice to bundle more than one response or request, or combine a request with a response, so the actual exchange may look different in practice to what is shown above.

## 4. Enhanced Telnet Option Negotiation

In order to accommodate the new environment option negotiations, the server will bundle an environment option invitation along with the standard terminal type invitation request to the client.

A client should either send a negative acknowledgment (WONT NEW-ENVIRON), or at some point after completing terminal type negotiations, but before completing the full set of negotiations required for transparent mode, engage in environment option sub-negotiation with the server.  A maximum or 1024 bytes of environment strings may be sent to the server.  A recommended sequence might look like the following:

```
AS/400 Telnet server              Enhanced Telnet client
--------------------------        -------------------------
IAC DO NEW-ENVIRON
IAC DO TERMINAL-TYPE        -->
(2 requests bundled)
                            <--   IAC WILL NEW-ENVIRON
IAC SB NEW-ENVIRON SEND
VAR IAC SE                  -->
                                  IAC SB NEW-ENVIRON IS
                                  VAR "USER" VALUE "JONES"
                                  USERVAR "DEVNAME" VALUE "MYDEVICE07"
                            <--   IAC SE
                            <--   IAC WILL TERMINAL-TYPE
                                  (do the terminal type
                                  sequence first)
IAC SB TERMINAL-TYPE SEND
IAC SE                      -->
                                  IAC SB TERMINAL-TYPE IS
                            <--   IBM-5555-C01 IAC SE
                                  (terminal type negotiations
                                  completed)
IAC DO EOR                  -->
(server will continue
with normal transparent
mode negotiations)
                            <--   IAC WILL EOR
                             .
                             .
  (other negotiations)       .
```

Actual bytes transmitted in the above example are shown in hex below.

```
   AS/400 Telnet server              Enhanced Telnet client
   --------------------------        -------------------------
   FF FD 27
   FF FD 18                     -->
   (2 requests bundled)
                                <--   FF FB 27
   FF FA 27 01 00 FF F0         -->
                                      FF FA 27 00 00 55 53 45
                                      52 01 4A 4F 4E 45 53 03
                                      44 45 56 4E 41 4D 45 01
                                      4D 59 44 45 56 49 43 45
                                <--   30 37 FF F0
                                <--   FF FB 18
                                      (do the terminal type
                                      sequence first)
   FF FA 18 01 FF F0            -->
                                      FF FA 18 00 49 42 4D 2D
                                      35 35 35 35 2D 43 30 31
                                <--   FF F0
   FF FD 19                     -->
   (server will continue
   with normal transparent
   mode negotiations)
                                <--   FF FB 19
                                 .
                                 .
   (other negotiations)          .
```

RFC 1572 defines 6 standard VAR's: USER, JOB, ACCT, PRINTER, SYSTEMTYPE, and DISPLAY.  The USER standard VAR will hold the value of the AS/400 user profile name to be used in auto-signon requests. The Telnet server will make no direct use of the additional 5 VAR's, nor are any of them required to be sent.  All standard VAR's and their values that are received by the Telnet server will be placed in a buffer, along with any USERVAR's received (described below), and made available to a registered initialization exit program to be used for any purpose desired.

There are some reasons you may want to send NEW-ENVIRON negotiations prior to TERMINAL-TYPE negotiations.  With AS/400 TELNET server, several virtual device modes can be negotiated: 1) VTxxx device 2) 3270 device 3) 5250 device (includes Network Station).  The virtual device mode selected depends on the TERMINAL-TYPE negotiated plus any other TELNET option negotiations necessary to support those modes. The AS/400 TELNET server will create the desired virtual device at the first opportunity it thinks it has all the requested attributes needed to create the device.  This can be as early as completion of the TERMINAL-TYPE negotiations.

For the case of Transparent mode (5250 device), then the moment TERMINAL-TYPE, BINARY, and EOR options are negotiated the TELNET server will go create the virtual device.  Receiving any NEW-ENVIRON negotiations after these option negotiations are complete will result in the NEW-ENVIRON negotiations having no effect on device attributes, as the virtual device will have already been created. So, for Transparent mode, NEW-ENVIRON negotiations are effectively closed once EOR is negotiated, since EOR is generally the last option done.

For other devices modes (such as VTxxx or 3270), you cannot be sure when the AS/400 TELNET server thinks it has all the attributes to create the device.  Recall that NEW-ENVIRON negotiations are optional, and therefore the AS/400 TELNET server need not wait for any NEW-ENVIRON options prior to creating the virtual device.  It is in the clients best interest to send NEW-ENVIRON negotiations as soon as possible, preferably before TERMINAL-TYPE is negotiated.  That way, the client can be sure the requested attributes were received before the virtual device is created.

## 5. Enhanced Display Emulation Support

RFC 1572 style USERVAR variables have been defined to allow a compliant Telnet client more control over the Telnet server virtual device on the AS/400. These USERVAR's allow the client Telnet to create or select a previously created virtual device. If the virtual device does not exist and must be created, then the USERVAR variables are used to create and initialize the device attributes. If the virtual device already exists, the device attributes are modified.

The USERVAR's defined to accomplish this are:

| USERVAR | VALUE | EXAMPLE | DESCRIPTION |
|---|---|---|---|
| DEVNAME | us-ascii char(x) | MYDEVICE07 | Display device name |
| KBDTYPE | us-ascii char(3) | USB | Keyboard type |
| CODEPAGE | us-ascii char(y) | 437 | Code page |
| CHARSET | us-ascii char(y) | 1212 | Character set |

x - up to a maximum of 10 characters
y - up to a maximum of 5 characters

For a description of the KBDTYPE, CODEPAGE and CHARSET parameters and their permissible values, refer to Chapter 8 in the Communications Configuration Reference [5] and also to Appendix C in National Language Support [16].

## 6. Enhanced Display Auto-Signon and Password Encryption

Several 5250 Telnet server specific USERVAR's will be defined. One will carry a random seed to be used in Data Encryption Standard (DES) password encryption, and another will carry the encrypted copy of the password. This would use the same password/substitution scheme as APPC and Client Access. For a description of the 7-step DES encryption scheme, refer to Federal Information Processing Standards Publication 46 [17] or visit the IBM Customer Support FTP Server at one of the following links:

ftp://ftp.networking.ibm.com/pub/standards/ciw/sig/sec/pwsubciw.ps
ftp://ftp.networking.ibm.com/pub/standards/ciw/sig/sec/pwsubciw.ps.Z
ftp://ftp.networking.ibm.com/pub/standards/ciw/sig/sec/pwsubciw.zip

If encrypted password exchange is not required, clear-text password exchange is permitted using the same USERVAR's defined for encryption. For this case, the random client seed should be set to either an empty value (RFC 1572 preferred method) or to hexadecimal zeros to indicate the password is not encrypted, but is clear-text.

It should be noted that security of clear-text password exchange cannot be guaranteed unless the network is physically protected or a trusted network (such as an intranet). If your network is vulnerable to IP address spoofing or directly connected to the Internet, you should engage in encrypted password exchange to validate a clients identity.

Additional VAR's and USERVAR's have also been defined to allow an auto-signon user greater control over their startup environment, similar to what is supported using the Open Virtual Terminal (QTVOPNVT) API [3].

The standard VAR's supported to accomplish this are:

| VAR | VALUE | EXAMPLE | DESCRIPTION |
|---|---|---|---|
| USER | us-ascii char(x) | USERXYZ | User profile name |

x - up to a maximum of 10 characters

The custom USERVAR's defined to accomplish this are:

| USERVAR | VALUE | EXAMPLE | DESCRIPTION |
|---|---|---|---|
| IBMRSEED | binary(8) | 8-byte hex field | Random client seed |
| IBMSUBSPW | binary(10) | 10-byte hex field | Substitute password |
| IBMCURLIB | us-ascii char(x) | QGPL | Current library |
| IBMIMENU | us-ascii char(x) | MAIN | Initial menu |
| IBMPROGRAM | us-ascii char(x) | QCMD | Program to call |

x - up to a maximum of 10 characters

In order to communicate the server random seed value to the client, the server will request a USERVAR name made up of a fixed part (the 8 characters "IBMRSEED" immediately followed by an 8-byte hexadecimal variable part, which is the server random seed.  The client generates its own 8-byte random seed value, and uses both seeds to encrypt the password.  Both the encrypted password and the client random seed value are then sent to the server for authentication.  RFC 1572 rules will need to be adhered to when transmitting the client random seed and substituted password values to the server.  Specifically, since a typical environment string is a variable length hexadecimal field, the hexadecimal fields are required to be escaped and/or byte stuffed according to the RFC 854 [8], where any single byte could be mis-construed as a Telnet IAC or other Telnet option negotiation control character.  The client must escape and/or byte stuff any bytes which could be seen as a RFC 1572 [13] option, specifically VAR, VALUE, ESC and USERVAR.

The following illustrates the encrypted case:

```
AS/400 Telnet server             Enhanced Telnet client
--------------------------       -------------------------------
IAC DO NEW-ENVIRON          -->
                            <--  IAC WILL NEW-ENVIRON
IAC SB NEW-ENVIRON SEND
USERVAR "IBMRSEEDxxxxxxxx"
USERVAR "IBMSUBSPW"
VAR USERVAR IAC SE          -->
                                 IAC SB NEW-ENVIRON IS
                                 VAR "USER" VALUE "SMITH"
                                 USERVAR "IBMRSEED" VALUE "yyyyyyyy"
                                 USERVAR "IBMSUBSPW" VALUE "zzzzzzzz"
                            <--  IAC SE
                             .
                             .
(other negotiations)         .
```

In this example, "xxxxxxxx" is an 8-byte hexadecimal random server seed, "yyyyyyyy" is an 8-byte hexadecimal random client seed and "zzzzzzzz" is an 8-byte hexadecimal encrypted password.  If the password is not valid, then the sign-on panel is displayed.  If the password is expired, then the Change Password panel is displayed.

Actual bytes transmitted in the above example are shown in hex below.

```
AS/400 Telnet server             Enhanced Telnet client
--------------------------        -------------------------
FF FD 27                    -->
                            <--  FF FB 27
FF FA 27 01 03 49 42 4D
52 53 45 45 44 78 78 78
78 78 78 78 78 03 49 42
4D 53 55 42 53 50 57 03
00 FF F0                    -->
                                 FF FA 27 00 00 55 53 45
                                 52 01 53 4D 49 54 48 03
                                 49 42 4D 52 53 45 45 44
                                 01 79 79 79 79 79 79 79
                                 79 03 49 42 4D 53 55 42
                                 53 50 57 01 7A 7A 7A 7A
                            <--  7A 7A 7A 7A FF F0
```

The following illustrates the clear-text case:

```
AS/400 Telnet server             Enhanced Telnet client
--------------------------        -------------------------
IAC DO NEW-ENVIRON          -->
                            <--  IAC WILL NEW-ENVIRON
IAC SB NEW-ENVIRON SEND
USERVAR "IBMRSEEDxxxxxxxx"
USERVAR "IBMSUBSPW"
VAR USERVAR IAC SE          -->
                                 IAC SB NEW-ENVIRON IS
                                 VAR "USER" VALUE "SMITH"
                                 USERVAR "IBMRSEED" VALUE
                                 USERVAR "IBMSUBSPW" VALUE "yyyyyyyy"
                            <--  IAC SE
                             .
                             .
(other negotiations)         .
```

In this example, "xxxxxxxx" is an 8-byte hexadecimal random server seed, "yyyyyyyyyy" is a 10-byte us-ascii client clear-text password. If the password has expired, then the sign-on panel is displayed.

Actual bytes transmitted in the above example are shown in hex below.

```
   AS/400 Telnet server             Enhanced Telnet client
   --------------------------        -------------------------
   FF FD 27                     -->
                                <--  FF FB 27
   FF FA 27 01 03 49 42 4D
   52 53 45 45 44 78 78 78
   78 78 78 78 78 03 49 42
   4D 53 55 42 53 50 57 03
   00 FF F0                     -->
                                     FF FA 27 00 00 55 53 45
                                     52 03 53 4D 49 54 48 03
                                     49 42 4D 52 53 45 45 44
                                     01 03 49 42 4D 53 55 42
                                     53 50 57 01 7A 7A 7A 7A
                                <--  7A 7A 7A 7A FF F0
```

## 7. Device Name Collision Processing

Device name collision occurs when a Telnet client sends the Telnet server a virtual device name that it wants to use, but that device is already in use on the server. When this occurs, the Telnet server sends a request to the client asking it to try another device name. The environment option negotiation uses the USERVAR name of DEVNAME to communicate the virtual device name. The following shows how the Telnet server will request the Telnet client to send a different DEVNAME when device name collision occurs.

```
   AS/400 Telnet server             Enhanced Telnet client
   --------------------------        -------------------------
   IAC SB NEW-ENVIRON SEND
   VAR USERVAR IAC SE         -->

   Server requests all environment variables be sent.

                                    IAC SB NEW-ENVIRON IS USERVAR
                                    "DEVNAME" VALUE "MYDEVICE1"
                                    USERVAR "xxxxx" VALUE "xxx"
                                    ...
                              <--   IAC SE

   Client sends all environment variables, including DEVNAME.  Server
   tries to select device MYDEVICE1.  If the device is already in use,
   server requests DEVNAME be sent again.

   IAC SB NEW-ENVIRON SEND
   USERVAR "DEVNAME" IAC SE   -->

   Server sends a request for a single environment variable: DEVNAME

                                    IAC SB NEW-ENVIRON IS USERVAR
                              <--   "DEVNAME" VALUE "MYDEVICE2" IAC SE
```

Client sends one environment variable, calculating a new value of MYDEVICE2. If MYDEVICE2 is different from the last request, then server tries to select device MYDEVICE2, else server disconnects client. If MYDEVICE2 is also in use, server will send DEVNAME request again, and keep doing so until it receives a device that is not in use, or the same device name twice in row.

## 8. Enhanced Printer Emulation Support

RFC 1572 style USERVAR variables have been defined to allow a compliant Telnet client more control over the Telnet server virtual device on the AS/400. These USERVAR's allow the client Telnet to select a previously created virtual device or auto-create a new virtual device with requested attributes.

This makes the enhancements available to any Telnet client that chooses to support the new negotiations.

The USERVAR's defined to accomplish this are:

| USERVAR | VALUE | EXAMPLE | DESCRIPTION |
|---|---|---|---|
| DEVNAME | us-ascii char(x) | PRINTER1 | Printer device name |
| IBMIGCFEAT | us-ascii char(6) | 2424J0 | IGC feature (DBCS) |
| IBMMSGQNAME | us-ascii char(x) | QSYSOPR | *MSGQ name |
| IBMMSGQLIB | us-ascii char(x) | QSYS | *MSGQ library |
| IBMFONT | us-ascii char(x) | 12 | Font |
| IBMFORMFEED | us-ascii char(1) | C \| U \| A | Formfeed |
| IBMBUFFERSIZE | us-ascii char(y) | 4096 | Reserved |
| IBMTRANSFORM | us-ascii char(1) | 1 \| 0 | Transform |
| IBMMFRTYPMDL | us-ascii char(x) | *IBM42023 | Mfg. type and model |
| IBMPPRSRC1 | binary(1) | 1-byte hex field | Paper source 1 |
| IBMPPRSRC2 | binary(1) | 1-byte hex field | Paper source 2 |
| IBMENVELOPE | binary(1) | 1-byte hex field | Envelope |
| IBMASCII899 | us-ascii char(1) | 1 \| 0 | ASCII 899 support |
| IBMWSCSTNAME | us-ascii char(x) | *NONE | WCS name |
| IBMWSCSTLIB | us-ascii char(x) | *LIBL | WCS library |
| IBMIGCFEAT | us-ascii char(6) | 2424J0 | IGC feature (DBCS) |

x - up to a maximum of 10 characters
y - up to a maximum of 5 characters

The "IBM" prefix on the USERVAR's denotes AS/400 specific attributes.

For a description of most of these parameters (drop the "IBM" from the USERVAR) and their permissible values, refer to Chapter 8 in the Communications Configuration Reference [5].

The IBMPPRSRC1, IBMPPRSRC2 and IBMENVELOPE custom USERVAR's do not map directly to their descriptions in Chapter 8 in the Communications Configuration Reference [5]. To map these, use the index listed here:

| IBMPPRSRC1 | HEX | IBMPPRSRC2 | HEX | IBMENVELOPE | HEX |
|---|---|---|---|---|---|
| *NONE | 'FF'X | *NONE | 'FF'X | *NONE | 'FF'X |
| *MFRTYPMDL | 'FE'X | *MFRTYPMDL | 'FE'X | *MFRTYPMDL | 'FE'X |
| *SAME | '00'X | *SAME | '00'X | *SAME | '00'X |
| *LETTER | '01'X | *LETTER | '01'X | *B5 | '06'X |
| *LEGAL | '02'X | *LEGAL | '02'X | *MONARCH | '09'X |
| *EXECUTIVE | '03'X | *EXECUTIVE | '03'X | *NUMBER9 | '0A'X |
| *A4 | '04'X | *A4 | '04'X | *NUMBER10 | '0B'X |
| *A5 | '05'X | *A5 | '05'X | *C5 | '0C'X |
| *B5 | '06'X | *B5 | '06'X | *DL | '0D'X |
| *CONT80 | '07'X | *CONT80 | '07'X | | |
| *CONT132 | '08'X | *CONT132 | '08'X | | |
| *A3 | '0E'X | *A3 | '0E'X | | |
| *B4 | '0F'X | *B4 | '0F'X | | |
| *LEDGER | '10'X | *LEDGER | '10'X | | |

Note 1: For IBMPPRSRC2, *CONT80 and *CONT132 support starts at V3R7.

Note 2: For IBMPPRSRC1 and IBMPPRSRC2, *A3, *B4 and *LEDGER support starts at V3R7.

## 9. Telnet Printer Terminal Types

New Telnet options are defined for the printer pass-through mode of operation.  To enable printer pass-through mode, both the client and server must agree to at least support the Transmit-Binary, End-Of-Record, and Terminal-Type Telnet options.  The following are new terminal types for printers:

| TERMINAL-TYPE | DESCRIPTION |
|---|---|
| IBM-5553-B01 | Double-Byte printer |
| IBM-3812-1 | Single-Byte printer |

Specific characteristics of the IBM-5553-B01 or IBM-3812-1 printers are specified through the USERVAR IBMMFRTYPMDL, which specifies the manufacturer type and model.

An example of a typical negotiation process to establish printer pass-through mode of operation is shown below.  In this example, the server initiates the negotiation by sending the DO TERMINAL-TYPE request.

```
   AS/400 Telnet server                Enhanced Telnet client
   --------------------------          --------------------------
   IAC DO NEW-ENVIRON         -->
                              <--   IAC WILL NEW-ENVIRON
   IAC SB NEW-ENVIRON SEND
   VAR USERVAR IAC SE         -->
                                    IAC SB NEW-ENVIRON IS
                                    USERVAR "DEVNAME" VALUE "PCPRINTER"
                                    USERVAR "IBMMSGQNAME" VALUE "QSYSOPR"
                                    USERVAR "IBMMSGQLIB" VALUE "*LIBL"
                                    USERVAR "IBMTRANSFORM" VALUE "0"
                                    USERVAR "IBMFONT" VALUE "12"
                                    USERVAR "IBMFORMFEED" VALUE "C"
                                    USERVAR "IBMBUFFERSIZE" VALUE "1024"
                                    USERVAR "IBMPPRSRC1" VALUE ESC '01'X
                                    USERVAR "IBMPPRSRC2" VALUE '04'X
                                    USERVAR "IBMENVELOPE" VALUE IAC 'FF'X
                              <--   IAC SE
   IAC DO TERMINAL-TYPE       -->
                              <--   IAC WILL TERMINAL-TYPE
   IAC SB TERMINAL-TYPE SEND
   IAC SE                     -->
                                    IAC SB TERMINAL-TYPE IS IBM-3812-1
                              <--   IAC SE
   IAC DO BINARY              -->
                              <--   IAC WILL BINARY
   IAC DO EOR                 -->
                              <--   IAC WILL EOR
```

Some points about the above example. The IBMPPRSRC1 value requires escaping the value using ESC according to RFC 1572 [13]. The IBMPPRSRC2 does not require an ESC character since '04'X has no conflict with RFC 1572 options. Finally, to send 'FF'X for the IBMENVELOPE value, escape the 'FF'X value by using another 'FF'X (called "doubling"), so as not to have the value interpreted as a Telnet character per RFC 854 [8].

Actual bytes transmitted in the above example are shown in hex below.

```
   AS/400 Telnet server             Enhanced Telnet client
   --------------------------       --------------------------
   FF FD 27                    -->
                               <--  FF FB 27
   FF FA 27 01 00 03 FF F0     -->
                                    FF FA 27 00 03 44 45 56
                                    4E 41 4D 45 01 50 43 50
                                    52 49 4E 54 45 52 03 49
                                    42 4D 4D 53 47 51 4E 41
                                    4D 45 01 51 53 59 53 4F
                                    50 52 03 49 42 4D 4D 53
                                    47 51 4C 49 42 01 2A 4C
                                    49 42 4C 03 49 42 4D 54
                                    52 41 4E 53 46 4F 52 4D
                                    01 30 03 49 42 4D 46 4F
                                    4E 54 01 31 32 03 49 42
                                    4D 46 4F 52 4D 46 45 45
                                    44 01 43 03 49 42 4D 42
                                    55 46 46 45 52 53 49 5A
                                    45 01 31 30 32 34 03 49
                                    42 4D 50 50 52 53 52 43
                                    31 01 02 01 03 49 42 4D
                                    50 50 52 53 52 43 32 01
                                    04 03 49 42 4D 45 4E 56
                                    45 4C 4F 50 45 01 FF FF
                               <--  FF F0
   FF FD 18                    -->
                               <--  FF FB 18
   FF FA 18 01 FF F0           -->
                                    FF FA 18 00 49 42 4D 2D
                               <--  33 38 31 32 2D 31 FF F0
   FF FD 00                    -->
                               <--  FF FB 00
   FF FD 19                    -->
                                    FF FB 19
```

## 10. Telnet Printer Startup Response Record for Printer Emulators

Once Telnet negotiation for a 5250 pass-through mode is completed, the 5250 Telnet server will initiate a virtual printer power-on sequence on behalf of the Telnet client. The Telnet server will supply a Startup Response Record to the Telnet client with the status of the printer power-on sequence, indicating success or failure of the virtual printer power-on sequence.

This section shows an example of two Startup Response Records. The source device is a type 3812 model 01 printer with name "PCPRINTER" on the target system "TARGET".

Figure 1 shows an example of a successful response; Figure 2 shows an example of an error response.

### 10.1 Example of a Success Response Record

The response record in Figure 1 was sent by an AS/400 at Release V4R2. It is an example of the target sending back a successful Startup Response Record.

```
   +--------------------------------------------------------------------+
   |       +-----  Pass-Through header                                  |
   |       |          +---  Response data                               |
   |       |          |            +----  Start diagnostic information  |
   |       |          |            |                                    |
   | +----------++----------++---------------------------------------   |
   | |          ||          ||                                          |
   | 004912A090000560060020C0003D0000C9F9F0F2E3C1D9C7C5E34040D7C3D7D9   |
   |                                 |      | T A R G E T     P C P R   |
   |                                 +------+                           |
   |                           Response Code (I902)                     |
   |                                                                    |
   | ----------------------------------------------------------------   |
   |                                                                    |
   | C9D5E3C5D9400000000000000000000000000000000000000000000000000000   |
   |  I N T E R                                                         |
   |                                                                    |
   |                  +------- End of diagnostic information            |
   |                  |                                                 |
   | -----------------+                                                 |
   |                  |                                                 |
   | 000000000000000000                                                 |
   +--------------------------------------------------------------------+
```

Figure 1. Example of a success response record.

- '0049'X = Length pass-through data, including this length field
- '12A0'X = GDS LU6.2 header
- '90000560060020C0003D0000'X = Fixed value fields
- 'C9F9F0F2'X                 = Response Code (I902)
- 'E3C1D9C7C5E34040'X         = System Name (TARGET)
- 'D7C3D7D9C9D5E3C5D940'X     = Object Name (PCPRINTER)

### 10.2 Example of an Error Response Record

The response record in Figure 2 is one that reports an error. The virtual device named "PCPRINTER", is not available on the target system "TARGET", because the device is not available. You would normally see this error if the printer was already assigned to another Telnet session.

```
   +--------------------------------------------------------------------+
   |       +-----  Pass-Through header                                  |
   |       |          +---  Response data                               |
   |       |          |            +----  Start diagnostic information  |
   |       |          |            |                                    |
   | +----------++----------++---------------------------------------   |
   | |          ||          ||                                          |
   | 004912A0900005600600820003D0000F8F9F0F2E3C1D9C7C5E34040D7C3D7D9   |
   |                                 |      | T A R G E T     P C P R   |
   |                                 +------+                           |
   |                           Response Code (8902)                     |
   |                                                                    |
   | ----------------------------------------------------------------   |
   |                                                                    |
   | C9D5E3C5D9400000000000000000000000000000000000000000000000000000   |
   |  I N T E R                                                         |
   |                                                                    |
   |                  +------- End of diagnostic information            |
   |                  |                                                 |
   | -----------------+                                                 |
   |                  |                                                 |
   | 000000000000000000                                                 |
   +--------------------------------------------------------------------+
```

Figure 2. Example of an error response record.

- '0049'X = Length pass-through data, including this length field
- '12A0'X = GDS LU6.2 header
- '90000560060020C0003D0000'X = Fixed value fields
- 'F8F9F0F2'X                 = Response Code (8902)
- 'E3C1D9C7C5E34040'X         = System Name (TARGET)
- 'D7C3D7D9C9D5E3C5D940'X     = Object Name (PCPRINTER)

### 10.3 Response Codes

The Start-Up Response Record success response codes:

| CODE | DESCRIPTION |
|---|---|
| I901 | Virtual device has less function than source device |
| I902 | Session successfully started |
| I906 | Automatic sign-on requested, but not allowed. Session still allowed; a sign-on screen will be coming. |

The Start-Up Response Record error response codes:

| CODE | DESCRIPTION |
|---|---|
| 2702 | Device description not found. |
| 2703 | Controller description not found. |
| 2777 | Damaged device description. |
| 8901 | Device not varied on. |
| 8902 | Device not available. |
| 8903 | Device not valid for session. |
| 8906 | Session initiation failed. |
| 8907 | Session failure. |
| 8910 | Controller not valid for session. |
| 8916 | No matching device found. |
| 8917 | Not authorized to object. |
| 8918 | Job canceled. |
| 8920 | Object partially damaged. |
| 8921 | Communications error. |
| 8922 | Negative response received. |
| 8923 | Start-up record built incorrectly. |
| 8925 | Creation of device failed. |
| 8928 | Change of device failed. |
| 8929 | Vary on or vary off failed. |
| 8930 | Message queue does not exist. |
| 8934 | Start-up for S/36 WSF received. |
| 8935 | Session rejected. |
| 8936 | Security failure on session attempt. |
| 8937 | Automatic sign-on rejected. |
| 8940 | Automatic configuration failed or not allowed. |
| I904 | Source system at incompatible release. |

## 11. Printer Steady-State Pass-Through Interface

The information in this section applies to the passthrough session after the receipt of startup confirmation records is complete.

Following is the printer header interface used by Telnet.

```
   +--------------------------------------------------------------------+
   |   +-- Length of structure (LLLL)                                   |
   |   |                                                                |
   |   |    +-- GDS identifier                                          |
   |   |    |                                                           |
   |   |    |    +-- Data flow record                                   |
   |   |    |    |                                                      |
   |   |    |    |   +-- Length of pass-through specific header (LL)    |
   |   |    |    |   |                                                  |
   |   |    |    |   |   +-- Flags                                      |
   |   |    |    |   |   |                                              |
   |   |    |    |   |   |   +-- Printer operation code                 |
   |   |    |    |   |   |   |                                          |
   |   |    |    |   |   |   |      +-- Diagnostic field - zero pad to  |
   |   |    |    |   |   |   |      |   LL specified                    |
   |   |    |    |   |   |   |      |                                   |
   |   |    |    |   |   |   |      |            +-- Printer data       |
   |   |    |    |   |   |   |      |            |                      |
   | +--+ +--+ +--+ ++ +--+ ++ +----------+ +----------------+          |
   | |  | |  | |  | || |  | || |          | |                |          |
   | xxxx 12A0 xxxx xx xxxx xx xxxxxxxxxxxx ... print data ...          |
   |                                                                    |
   +--------------------------------------------------------------------+
    Figure 3. Layout of the printer pass-through header
```

BYTES 0-1:   Length of structure including this field (LLLL)

BYTES 2-3:   GDS Identifier ('12A0'X)

BYTE 4-5: Data flow record

This field contains flags that describe what type of data pass-through should expect to find following this header. Generally, bits 0-2 in the first byte are mutually exclusive (that is, if one of them is set to '1'B, the rest will be set to '0'B.) The bits, and their meanings follow.

| BIT | DESCRIPTION |
|---|---|
| 0 | Start-Up confirmation |
| 1 | Termination record |
| 2 | Start-Up Record |
| 3 | Diagnostic information included |
| 4 - 5 | Reserved |
| 6 | Reserved |
| 7 | Printer record |
| 8 - 13 | Reserved |
| 14 | Client-originated (inbound) printer record |
| 15 | Server-originated (outbound) printer record |

BYTE 6: Length printer pass-through header including this field (LL)

BYTES 7-8: Flags

```
  BYTE 7 BITS:  xxxx x111 --> Reserved
                xxxx 1xxx --> Last of chain
                xxx1 xxxx --> First of chain
                xx1x xxxx --> Printer now ready
                x1xx xxxx --> Intervention Required
                1xxx xxxx --> Error Indicator

  BYTE 8 BITS:  xxxx xxxx --> Reserved
```

BYTE 9: Printer operation code

| | |
|---|---|
| '01'X | Print/Print complete |
| '02'X | Clear Print Buffers |

BYTE 10-LL:  Diagnostic information (1)

If BYTE 7 = x1xx xxxx then bytes 10-LL may contain: (2)

| | |
|---|---|
| Printer not ready | C9 00 03 02 51 00 |
| End of forms | C9 00 03 02 50 00 |
| Graphic check | C9 00 03 02 26 00 |
| Data stream exception | C9 00 03 02 66 00 |
| Data stream exception | C9 00 03 02 67 00 |
| Data stream exception | C9 00 03 02 68 00 |
| Data stream exception | C9 00 03 02 69 00 |

If BYTE 7 = 1xxx xxxx then bytes 10-LL may contain: (3)

| | |
|---|---|
| Cancel | 08 11 02 00 00 00 |
| Invalid print parameter | 08 11 02 29 00 00 |
| Invalid print command | 08 11 02 28 00 00 |

Figure notes:

1. LL is the length of the structure defined in Byte 6.  If no additional data is present, the remainder of the structure must be padded with zeroes.

2. These are printer signal commands.  Further information on these commands may be obtained from the 5494 Remote Control Unit Functions Reference guide [2].  Refer to your printer documentation for more specific information on these data stream exceptions.

3. These are printer negative responses.  Further information on these commands may be obtained from the 5494 Remote Control Unit Functions Reference guide [2].

The print data will start in byte LL+1.

## 11.1 Example of a Print Record

Figure 4 shows the server sending the client data with a print record.  This is normally seen following receipt of a Success Response Record, such as the example in Figure 1.

```
+--------------------------------------------------------------------+
|   +-- Length of structure (LLLL)                                   |
|   |    +-- GDS identifier                                          |
|   |    |    +-- Data flow record                                   |
|   |    |    |   +-- Length of pass-through specific header (LL)    |
|   |    |    |   |   +-- Flags                                      |
|   |    |    |   |   |   +-- Printer operation code                 |
|   |    |    |   |   |   |      +-- Zero pad to LL specified (0A)   |
|   |    |    |   |   |   |      |            +-- Printer data       |
|   |    |    |   |   |   |      |            |                      |
| +--+ +--+ +--+ ++ +--+ ++ +----------+ +---------------------------|
| |  | |  | |  | || |  | || |          | |                           |
| 0085 12A0 0101 0A 1800 01 000000000000 34C4012BD20345FF2BD2044C0002|
|                                                                    |
| ------------------------------------------------------------       |
|                                                                    |
| 2BD2040D00002BD20A8501010201030204022BD20309022BD2061100014A       |
|                                                                    |
| ------------------------------------------------------------       |
|                                                                    |
| 402BD20601010000012BD306F60000FFFF2BD20A48000001000000010100       |
|                                                                    |
| ------------------------------------------------------------       |
|                                                                    |
| 2BD10705000B0090012BD2044900F02BD206404A403DE02BD2041500F034       |
|                                                                    |
|    end of printer data                                             |
| -------------------------+                                         |
|                          |                                         |
| C4012BD10381FF002BC8034001                                         |
+--------------------------------------------------------------------+
```

Figure 4. Server sending client data with a print record

- '0085'X         = Logical record length, including this byte (LLLL)
- '12A0'X         = GDS LU6.2 header
- '0101'X         = Data flow record (server to client)
- '0A'X           = Length of pass-through specific header (LL)
- '1800'X         = First of chain / Last of chain indicators
- '01'X           = Print
- '000000000000'X = Zero pad header to LL specified
- '34C401'X       = First piece of data for spooled data
- Remainder is printer data/commands/orders

## 11.2 Example of a Print Complete Record

Figure 5 shows the client sending the server a print complete record. This would normally follow receipt of a print record, such as the example in Figure 4.  This indicates successful completion of a print request.

```
   +---------------------------------------------------------------------+
   |   +-- Length of structure (LLLL)                                    |
   |   |    +-- GDS identifier                                           |
   |   |    |    +-- Data flow record                                    |
   |   |    |    |   +-- Length of pass-through specific header (LL)     |
   |   |    |    |   |   +-- Flags                                       |
   |   |    |    |   |   |   +-- Printer operation code                  |
   |   |    |    |   |   |   |                                           |
   | +--+ +--+ +--+ ++ +--+ ++                                           |
   | |  | |  | |  | || |  | ||                                           |
   | 000A 12A0 0102 04 0000 01                                           |
   +---------------------------------------------------------------------+
```

Figure 5. Client sending server a print complete record

- '000A'X = Logical record length, including this byte (LLLL)
- '12A0'X = GDS LU6.2 header
- '0102'X = Data flow response record (client to server)
- '04'X   = Length of pass-through specific header (LL)
- '0000'X = Good Response
- '01'X   = Print Complete

## 11.3 Example of a Null Print Record

Figure 6 shows the server sending the client a null print record. The null print record is the last print command the server sends to the client for a print job, and indicates to the printer there is no more data.

This example would normally follow any number of print records, such as the example in Figure 4. This indicates successful completion of a print job. The client normally responds to this null print record with another print complete record, such as in Figure 5.

```
+--------------------------------------------------------------------+
|   +-- Length of structure (LLLL)                                   |
|   |    +-- GDS identifier                                          |
|   |    |    +-- Data flow record                                   |
|   |    |    |   +-- Length of pass-through specific header (LL)    |
|   |    |    |   |   +-- Flags                                      |
|   |    |    |   |   |   +-- Printer operation code                 |
|   |    |    |   |   |   |      +-- Zero pad to LL specified (0A)   |
|   |    |    |   |   |   |      |        +-- Printer data           |
|   |    |    |   |   |   |      |        |                          |
| +--+ +--+ +--+ ++ +--+ ++ +----------+ ++                          |
| |  | |  | |  | || |  | || |          | ||                          |
| 0011 12A0 0101 0A 0800 01 000000000000 00                          |
+--------------------------------------------------------------------+
```

Figure 6. Server sending client a null print record

- '0011'X         = Logical record length, including this byte
- '12A0'X         = GDS LU6.2 header
- '0101'X         = Data flow record
- '0A'X           = Length of pass-through specific header (LL)
- '0800'X         = Last of Chain
- '01'X           = Print
- '000000000000'X = Zero pad header to LL specified
- '00'X           = Null data byte

## 12. End-to-End Print Example

The next example shows a full print exchange between a Telnet client and server for a 526 byte spooled file.  Selective translation of the hexadecimal streams into 1) Telnet negotiations and 2) ASCII/EBCDIC characters are done to aid readability.  Telnet negotiations are delimited by '(' and ')' parenthesis characters; ASCII/EBCDIC conversions are bracketed by '|' vertical bar characters.

```
   AS/400 Telnet server               Enhanced Telnet client
   ---------------------------------  ---------------------------------
   FFFD27                         -->

   (IAC DO NEW-ENVIRON)
                                  <-- FFFB27

                                      (IAC WILL NEW-ENVIRON)

   FFFD18FFFA270103 49424D5253454544
   7CF9630A63D18004 0003FFF0      -->

   (IAC DO TERMINAL-TYPE
   IAC SB NEW-ENVIRON SEND USERVAR
   IBMRSEED xxxxxxxx VAR USERVAR
   IAC SE)

                                      FFFB18FFFA270003 49424D5253454544
                                      7CF9630A63D18004 00034445564E414D
                                      450144554D4D5950 52540349424D4D53
                                      47514E414D450151 5359534F50520349
                                      424D4D5347514C49 42012A4C49424C03
                                      49424D464F4E5401 3031310349424D46
                                      4F524D4645454401 0349424D42554646
                                      455253495A450137 36380349424D5452
                                  <-- 414E53464F524D01 30FFF0

                                      (IAC WILL TERMINAL-TYPE IAC SB
                                       NEW-ENVIRON IS USERVAR
                                       IBMRSEED xxxxxxxx VAR
                                       USERVAR DEVNAME VALUE DUMMYPRT
                                       USERVAR IBMMSGQNAME VALUE
                                       USERVAR IBMMSGQLIB VALUE *LIBL
                                       USERVAR IBMFONT VALUE 011
                                       USERVAR IBMFORMFEED VALUE
                                       USERVAR IBMBUFFERSIZE VALUE 768
                                       USERVAR IBMTRANSFORM VALUE 0
                                       IAC SE)
```

```
   FFFA1801FFF0                      -->

   (IAC DO SB TERMINAL-TYPE SEND
   IAC SE)
                                     <-- FFFA180049424D2D 333831322D31FFF0

                                         (IAC SB TERMINAL-TYPE IS
                                          IBM-3812-1 IAC SE)

   FFFD19                            -->

   (IAC DO EOR)
                                     <-- FFFB19

                                         (IAC WILL EOR)

   FFFB19FFFD00FFFB 00               -->

   (IAC WILL EOR IAC DO BINARY
   IAC WILL BINARY)
                                     <-- FFFD19FFFB00FFFD 00

                                         (IAC DO EOR IAC WILL BINARY
                                          IAC DO BINARY)

   FFFD00                            -->

   (IAC DO BINARY)

   FFFB00004912A090 000560060020C000     |..........-...{.|
   3D0000C9F9F0F2C1 E2F4F0F040404044     |...I902AS400   D| (EBCDIC)
   554D4D5950525440 4000000000000000     |UMMYPRT         |
   0000000000000000 0000000000000000     |................|
   0000000000000000 00000000FFEF     --> |..............  |

   (IAC WILL BINARY ... 73-byte
    startup success response
    record ... IAC EOR)
```

```
   007812A001010A18 0001000000000000     |.x..............|
   036611180D12141B 461B481B4F1B541B     |.f......F.H.O.T.| (ASCII)
   55001B57001B3500 1B5F001B2D001B36     |U..W..5.._..-..6|
   1B49021B461B481B 2D001B410C1B321B     |.I..F.H.-..A..2.|
   57001B461B49121B 57001B461B49121B     |W..F.I..W..F.I..|
   461B481B2D001B41 0C1B321B57001B46     |F.H.-..A..2.W..F|
   1B49121B57001B46 1B49121B57001B46     |.I..W..F.I..W..F|
   1B49121B410C1B32 FFEF             --> |.I..A..2..      |

   (... 120-byte print record ...
    ... first of chain ...
    ... last of chain ... IAC EOR)

                                     <-- 000A12A001020400 0001FFEF

                                         (10-byte print complete header)

   008B12A001010A18 0001000000000000     |................|
   03241B57001B461B 49121B410C1B321B     |.$.W..F.I..A..2.| (ASCII)
   43421B410C1B321B 410C1B320A0A0A0A     |CB.A..2.A..2....|
   0A1B410C1B320350 2020202020202020     |..A..2.P        |
   2020202020202020 2020202020202050     |               P|
   72696E74204B6579 204F757470757420     |rint Key Output |
   2020202020202020 2020202020202020     |                |
   2020202020202020 2020202020202050     |               P|
   616765202020310D 03010DFFEF       --> |age   1......   |

   (... 139-byte print record ...
    ... first of chain ...
    ... last of chain ... IAC EOR)

                                     <-- 000A12A001020400 0001FFEF

                                         (10-byte print complete header)

   031012A001010A10 0001000000000000     |................|
   03010C03241B5700 1B461B49121B410C     |....$.W..F.I..A.| (ASCII)
   1B321B43421B410C 1B321B410C1B320A     |.2.CB.A..2.A..2.|
   0A0A0A0A1B410C1B 3203502020202020     |.....A..2.P     |
   2020202020202020 2020202020202020     |                |
   20205072696E7420 4B6579204F757470     |  Print Key Outp|
   7574202020202020 2020202020202020     |ut              |
   2020202020202020 2020202020202020     |                |
   2020506167652020 20310D03010D03FF     |  Page   1......|
   FF0A202020203537 3639535331205634     |..    5769SS1 V4|
   52324D3020393830 3232382020202020     |R2M0 980228     |
   2020202020202020 2020204153343030     |           AS400|
   2020202020202020 2020202020202030     |               0|
```

```
   342F32392F393820 2031363A30393A30     |4/29/98  16:09:0|
   340D0A0D0A202020 20446973706C6179     |4....    Display|
   2044657669636520 202E202E202E202E     | Device  . . . .|
   202E203A20205150 4144455630303130     | . :  QPADEV0010|
   0D0A202020205573 657220202E202E20     |..    User  . . |
   2E202E202E202E20 2E202E202E202E20     |. . . . . . . . |
   3A202044554D4D59 5553520D0A0D0A20     |:  DUMMYUSR.... |
   2020202020202020 2020202020202020     |                |
   2020202020202020 2020202020202020     |                |
   436F6D6D616E6420 456E747279202020     |Command Entry   |
   2020202020202020 2020202020202020     |                |
   2020202020204153 3430300D0A202020     |      AS400..   |
   03FFFF2020202020 2020202020202020     |...             |
   2020202020202020 2020202020202020     |                |
   2020202020202020 2020202020202020     |                |
   310D0A2050726576 696F757320636F6D     |1.. Previous com|
   6D616E647320616E 64206D6573736167     |mands and messag|
   65733A0D0A0D0A20 2020284E6F207072     |es:....   (No pr|
   6576696F75732063 6F6D6D616E647320     |evious commands |
   6F72206D65737361 676573290D0A0D0A     |or messages)....|
   0D0A0D0A0D0A0D0A 0D0A0D0A0D0A0D0A     |................|
   0D0A202020202020 2020202020202020     |..              |
   2020202020202020 2020202020202020     |                |
   2020202020202020 2020202020202020     |                |
   2020202020202020 2020202020202020     |                |
   2020202020202020 202020426F74746F     |           Botto|
   6D0D0A20547903D6 706520636F6D6D61     |m.. Ty..pe comma|
   6E642C2070726573 7320456E7465722E     |nd, press Enter.|
   0D0A203D3D3D3E20 53616D706C652050     |.. ===> Sample P|
   72696E7420536372 65656E2E2E2E0D0A     |rint Screen.....|
   0D0A0D0A0D0A2046 333D457869742020     |...... F3=Exit  |
   2046343D50726F6D 707420202046393D     | F4=Prompt   F9=|
   5265747269657665 2020204631303D49     |Retrieve   F10=I|
   6E636C7564652064 657461696C656420     |nclude detailed |
   6D6573736167FFEF                  --> |messag..        |

   (... 784-byte print record ...
    ... first of chain ... IAC EOR)

                                     <-- 000A12A001020400 0001FFEF

                                         (10-byte print complete header)
```

```
   006812A001010A00 0001000000000000     |.h..............|
   65730D0A20463131 3D446973706C6179     |es.. F11=Display| (ASCII)
   2066756C6C202020 2020204631323D43     | full      F12=C|
   616E63656C202020 204631333D496E66     |ancel    F13=Inf|
   6F726D6174696F6E 2041737369737461     |ormation Assista|
   6E74202020463234 3D4D6F7265206B65     |nt   F24=More ke|
   79730D0A0D0A0D0C FFEF             --> |ys........      |

   (... 104-byte print record ...
    IAC EOR)

                                     <-- 000A12A001020400 0001FFEF

                                         (10-byte print complete header)

   002912A001010A00 0001000000000000     |.)..............|
   03171B461B481B4F 1B541B55001B5700     |...F.H.O.T.U..W.| (ASCII)
   1B35001B5F001B2D 00FFEF           --> |.5.._..-...     |

   (... 41-byte print record ...
    IAC EOR)

                                     <-- 000A12A001020400 0001FFEF

                                         (10-byte print complete header)

   001112A001010A08 0001000000000000
   00FFEF                            -->

   (... 17-byte NULL print record ...
    ... last of chain ... IAC EOR)

                                     <-- 000A12A001020400 0001FFEF

                                         (10-byte print complete header)
```

## 13. Author's Note

Discussion of this draft should occur in one of these mailing lists:

TN3270E List (Roger Fajman raf@cu.nih.gov). Send subscription requests as e-mail with "subscribe tn3270e your_full_name" to listserv@list.nih.gov.

Midrange-L List (David Gibbs david@midrange.com). Send subscription requests as email with "subscribe midrange-l your_internet_address" to majordomo@midrange.com.

Telnet Working Group Mailing List: Send subscription requests as email with "subscribe telnet-ietf" to telnet-ietf-request@bsdi.com.

## 14. References

[1] IBM, "IBM 5250 Information Display System, Functions Reference Manual", SA21-9247-6, March 1987.

[2] IBM, "5494 Remote Control Unit, Functions Reference", SC30-3533-03, November 1994.

[3] IBM, "AS/400 System API Reference", SC41-5801, February 1998.

[4] IBM, "AS/400 TCP/IP Configuration and Reference", SC41-5420-01, February 1998.

[5] IBM, "AS/400 Communications Configuration", SC41-5401-00, August 1997.

[6] IBM, "SNA Formats", GA27-3136-13, November 1993.

[7] IBM, "Using the Pageprinter 3812 with System/36 or System/38", S544-3343-01, September 1997.

[8] Postel, J. and J. Reynolds, "TELNET PROTOCOL SPECIFICATION", RFC 854, USC/Information Sciences Institute, May 1983.

[9] Postel, J. and J. Reynolds, "TELNET OPTION SPECIFICATIONS", RFC 855, USC/Information Sciences Institute, May 1983.

[10] Postel, J. and J. Reynolds, "TELNET BINARY TRANSMISSION", RFC 856, USC/Information Sciences Institute, May 1983.

[11] VanBokkeln, J., "Telnet Terminal-Type Option", RFC 1091, FTP

Software, Inc., February 1989.

[12]  Postel, J. and J. Reynolds, "TELNET END OF RECORD OPTION", RFC 885, USC/Information Sciences Institute, December 1983.

[13]  Alexander, S., "Telnet Environment Option", RFC 1572, Lachman Technology, Inc., January 1994.

[14]  Chmielewski, P., "5250 Telnet Interface", RFC 1205, IBM Corporation, February 1991.

[15]  Postel, J. and J. Reynolds, "TELNET SUPPRESS GO AHEAD OPTION", RFC 858, Information Sciences Institute, May 1983.

[16]  IBM, "AS/400 National Language Support", SC41-5101-01, February 1998.

[17]  Data Encryption Standard, Federal Information Processing Standards Publication 46, January 15, 1977.

[18] Reynolds, J., and J. Postel, "Assigned Numbers", RFC 1700, STD 2, USC/Information Sciences Institute, October 1994.

## 15. Security Considerations

Security considerations of passwords are discussed in Section 6.

## 16. Author's Address

| | |
|---|---|
| Thomas E. Murphy, Jr. | Phone:  (607) 752-5482 |
| IBM Corporation | Fax:    (607) 752-5421 |
| 1701 North Street | Email:  murphyte@us.ibm.com |
| Endicott, NY 13760 | |
| Paul F. Rieth | Phone:  (607) 752-5474 |
| IBM Corporation | Fax:    (607) 752-5421 |
| 1701 North Street | Email:  rieth@us.ibm.com |
| Endicott, NY 13760 | |
| Jeffrey S. Stevens | Phone:  (607) 752-5488 |
| IBM Corporation | Fax:    (607) 752-5421 |
| 1701 North Street | Email:  jssteven@us.ibm.com |
| Endicott, NY 13760 | |

## 17. Relation to Other RFC's

UPDATES

This draft is an update to RFC 1205 [14], which describes the 5250 Telnet Interface. This update enhances that description to include device negotiation as well as printer support.

This draft makes use of RFC 1572 [13] to enhance communications with 5250 Telnet clients. RFC 1572 is currently on the Standards Track as a Proposed Standard, and is listed in Assigned Numbers [18].