Network Working Group
Request for Comments: 2877
Category: Informational
Updates: 1205

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

July 2000

5250 Telnet Enhancements

Status of this Memo

Copyright Notice

Abstract

This memo 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 memo 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.

Table of Contents

LIST OF FIGURES

1. 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.

2. 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.

3. 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 5250 transparent mode, engage in environment option sub-negotiation with the server. A maximum of 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.

4. 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].

The CODEPAGE and CHARSET USERVAR's must be associated with a KBDTYPE USERVAR. If either CODEPAGE or CHARSET are sent without KBDTYPE, they will default to system values. A default value for KBDTYPE can be sent to force CODEPAGE and CHARSET values to be used.

AS/400 system objects such as device names, user profiles, clear-text passwords, programs, libraries, etc. are required to be specified in English Upper Case (EUC). This includes:

Any letter (A-Z), any number (0-9), special characters (# $ _ @)

Therefore, where us-ascii is specified for VAR or USERVAR values, it is recommended that upper-cased ASCII values be sent, which will be converted to EBCDIC by the Telnet server.

A special case occurs for encrypted passwords (described in the next section), where both the initial password and user profile used to build the encrypted password must be EBCDIC English Upper Case, in order to be properly authenticated by the Telnet server.

5. 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 7-step DES-based password substitution scheme as APPC and Client Access. For a description of DES encryption, refer to Federal Information Processing Standards Publications (FIPS) 46-2 [17] and 81 [18], which can be found at the Federal Information Processing Standards Publications link:

http://www.itl.nist.gov/div897/pubs/by-num.htm

For a description of the 7-step password substitution scheme, refer to these IBM Customer Support FTP Server 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 "DUMMYUSR"
                                  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, where the server seed is "7D3E488F18080404", the client seed is "4E4142334E414233" and the encrypted password is "DFB0402F22ABA3BA". The user profile used to generate the encrypted password is "44554D4D59555352" (DUMMYUSR), with a clear-text password of "44554D4D595057" (DUMMYPW).

```
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 7D 3E 48
8F 18 08 04 04 03 49 42
4D 53 55 42 53 50 57 03
00 FF F0                    -->
                                  FF FA 27 00 00 55 53 45
                                  52 01 44 55 4D 4D 59 55
                                  53 52 03 49 42 4D 52 53
                                  45 45 44 01 4E 41 42 33
                                  4E 41 42 33 03 49 42 4D
```

```
                                    53 55 42 53 50 57 01 DF
                                    B0 40 2F 22 AB A3 BA 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 "DUMMYUSR"
                                     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, where the server seed is "7D3E488F18080404", the client seed is empty and the clear-text password is "44554D4D595057" (DUMMYPW). The user profile used is "44554D4D59555352" (DUMMYUSR).

```
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 7D 3E 48
8F 18 08 04 04 03 49 42
4D 53 55 42 53 50 57 03
00 FF F0                        -->
                                     FF FA 27 00 00 55 53 45
                                     52 01 44 55 4D 4D 59 55
                                     53 52 03 49 42 4D 52 53
                                     45 45 44 01 03 49 42 4D
                                     53 55 42 53 50 57 01 44
                                <--  55 4D 4D 59 50 57 FF F0
```

5.1 Password Substitutes Processing

Both APPC and Client Access use well-known DES encryption algorithms to create encrypted passwords. A Network Station or Enhanced Client can generate compatible encrypted passwords if they follow these steps, details of which can be found in the Federal Information Processing Standards 46-2 [17].

1. Padded_PW = Left justified user password padded to the right with ’40’X to 8 bytes.

   The users password must be left justified in an 8 byte variable and padded to the right with ’40’X up to an 8 byte length. If the users password is 8 bytes in length, no padding would occur. For computing password substitutes for passwords of length 9 and 10 see section "Handling passwords of length 9 and 10" below. Passwords less than 1 byte or greater than 10 bytes in length are not valid. Please note, if password is not in EBCDIC, it must be converted to EBCDIC uppercase.

2. XOR_PW = Padded_PW xor ’5555555555555555’X

   The padded password is Exclusive OR’ed with 8 bytes of ’55’X.

3. SHIFT_RESULT = XOR_PW << 1

   The entire 8 byte result is shifted 1 bit to the left; the leftmost bit value is discarded, and the rightmost bit value is cleared to 0.

4. PW_TOKEN = DES_ECB_mode(SHIFT_RESULT, /* key */
   userID_in_EBCDIC_uppercase /* data */)

   This shifted result is used as key to the Data Encryption Standard (Federal Information Processing Standards 46-2 [17]) to encipher the user identifier. When the user identifier is less than 8 bytes, it is left justified in an 8 byte variable and padded to the right with ’40’X. When the user identifier is 9 or 10 bytes, it is first padded to the right with ’40’X to a length of 10 bytes. Then bytes 9 and 10 are "folded" into bytes 1-8 using the following algorithm:

   Bit 0 is the high-order bit (i.e. has value of ’80’X).

   Byte 1, bits 0 and 1 are replaced with byte 1, bits 0 and 1 Exclusive OR’ed with byte 9, bits 0 and 1.
   Byte 2, bits 0 and 1 are replaced with byte 2, bits 0 and 1 Exclusive OR’ed with byte 9, bits 2 and 3.

Byte 3, bits 0 and 1 are replaced with byte 3, bits 0 and 1 Exclusive OR'ed with byte 9, bits 4 and 5.
Byte 4, bits 0 and 1 are replaced with byte 4, bits 0 and 1 Exclusive OR'ed with byte 9, bits 6 and 7.
Byte 5, bits 0 and 1 are replaced with byte 5, bits 0 and 1 Exclusive OR'ed with byte 10, bits 0 and 1.
Byte 6, bits 0 and 1 are replaced with byte 6, bits 0 and 1 Exclusive OR'ed with byte 10, bits 2 and 3.
Byte 7, bits 0 and 1 are replaced with byte 7, bits 0 and 1 Exclusive OR'ed with byte 10, bits 4 and 5.
Byte 8, bits 0 and 1 are replaced with byte 8, bits 0 and 1 Exclusive OR'ed with byte 10, bits 6 and 7.

User identifier greater than 10 bytes or less than 1 byte are not the result of this encryption id known as PW_TOKEN in the paper.

5. Increment PWSEQs and store it.

   Each LU must maintain a pair of sequence numbers for ATTACHs sent and received on each session. Each time an ATTACH is generated, (and password substitutes are in use on the session) the sending sequence number, PWSEQs, is incremented and saved for the next time. Both values are set to zero at BIND time. So the first use of PWSEQs has the value of 1, and increases by one with each use. A new field is added to the ATTACH to carry this sequence number. However, in certain error conditions, it is possible for the sending side to increment the sequence number and the receiver may not increment it. When the sender sends a subsequent ATTACH, the receiver will detect a missing sequence. This is allowed. However the sequence number received must always be larger than the previous one, even if some are missing.

   The maximum number of consecutive missing sequence numbers allowed is 16. If this is exceeded, the session is unbound with a protocol violation.

   Note: The sequence number must be incremented for every ATTACH sent. However, the sequence number field is only required to be included in the FMH5 if a password substitute is sent (byte 4, bit 3 on).

6. RDrSEQ = RDr + PWSEQs /* RDr is server seed. */

   The current value of PWSEQs is added to RDr, the random value received from the partner LU on this session, yielding RDrSEQ, essentially a predictably modified value of the random value received from the partner LU at BIND time.

```
7. PW_SUB = DES_CBC_mode(PW_TOKEN,        /* key      */
                         (RDrSEQ,         /* 8 bytes  */
                          RDs,            /* 8 bytes  */
                          ID xor RDrSEQ,  /* 16 bytes */
                          PWSEQs,         /* 8 bytes  */
                          )               /* data     */
                         )
```

The PW_TOKEN is used as a key to the DES function to generate a 8 bytes value for the following string of inputs. The DES CBC mode Initialization Vector (IV) used is 8 bytes of '00'X.

RDrSEQ: the random data value received from the partner LU plus the sequence number.

RDs: the random data value sent to the partner LU on BIND for this session.

A 16 byte value created by:

- padding the user identifier with '40'X to a length of 16 bytes.

- Exclusive OR the two 8 byte halves of the padded user identifier with the RDrSEQ value.

Note: User ID must first be converted to EBCDIC upper case.

PWSEQs: the sequence number.

This is similar to the process used on LU-LU verification as described in the Enhanced LU-LU Bind Security. The resulting enciphered random data is the 'password substitute'.

5.2 Handling passwords of length 9 and 10

1. Generate PW_TOKENa by using characters 1 to 8 of the password and steps 1-4 from the previous section.

2. Generate PW_TOKENb by using characters 9 and 10 and steps 1-4 from the previous section. In this case Padded_PW from step 1 will be characters 9 and 10 padded to the right with '40'X, for a total length of 8.

3. PW_TOKEN = PW_TOKENa xor PW_TOKENb

4. Now compute PW_SUB by performing steps 5-7 from the previous section.

5.3 Example Password Substitute Calculation

```
ID:           USER123
Password:     ABCDEFG
Server seed:  '7D4C2319F28004B2'X
Client seed:  '08BEF662D851F4B1'X
PWSEQs:       1     (PWSEQs is a sequence number needed in the
                      7-step encryption, and it is always one)
```

Encrypted Password should be : '5A58BD50E4DD9B5F'X

6. 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.

7. 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 chonoses 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
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 hopper
IBMASCII899	us-ascii char(1)	1 | 0	ASCII 899 support
IBMWSCSTNAME	us-ascii char(x)	*NONE	WSCST name
IBMWSCSTLIB	us-ascii char(x)	*LIBL	WSCST library

x - up to a maximum of 10 characters

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

The DEVNAME USERVAR is used both for displays and printers. The IBMFONT and IBMASCII899 are used only for SBCS environments.

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 IBMIGCFEAT supports the following variable DBCS language identifiers in position 5 (positions 1-4 must be '2424', position 6 must be '0'):

'J' = Japanese 'K' = Korean
'C' = Traditional Chinese 'S' = Simplified Chinese

The IBMTRANSFORM and IBMASCII899 values correspond to:

'1' = Yes '2' = No

The IBMFORMFEED values correspond to:

'C' = Continuous 'U' = Cut 'A' = Autocut

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	'00'X	*MFRTYPMDL	'00'X	*MFRTYPMDL	'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.

8. 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.

For DBCS environments, if IBMTRANSFORM is set to 1 (use Host Print Transform), then the virtual device created is 3812, not 5553. Therefore, IBM-3812-1 should be negotiated for TERMINAL-TYPE, and not IBM-5553-B01.

```
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 "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 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

9. 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.

9.1 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.

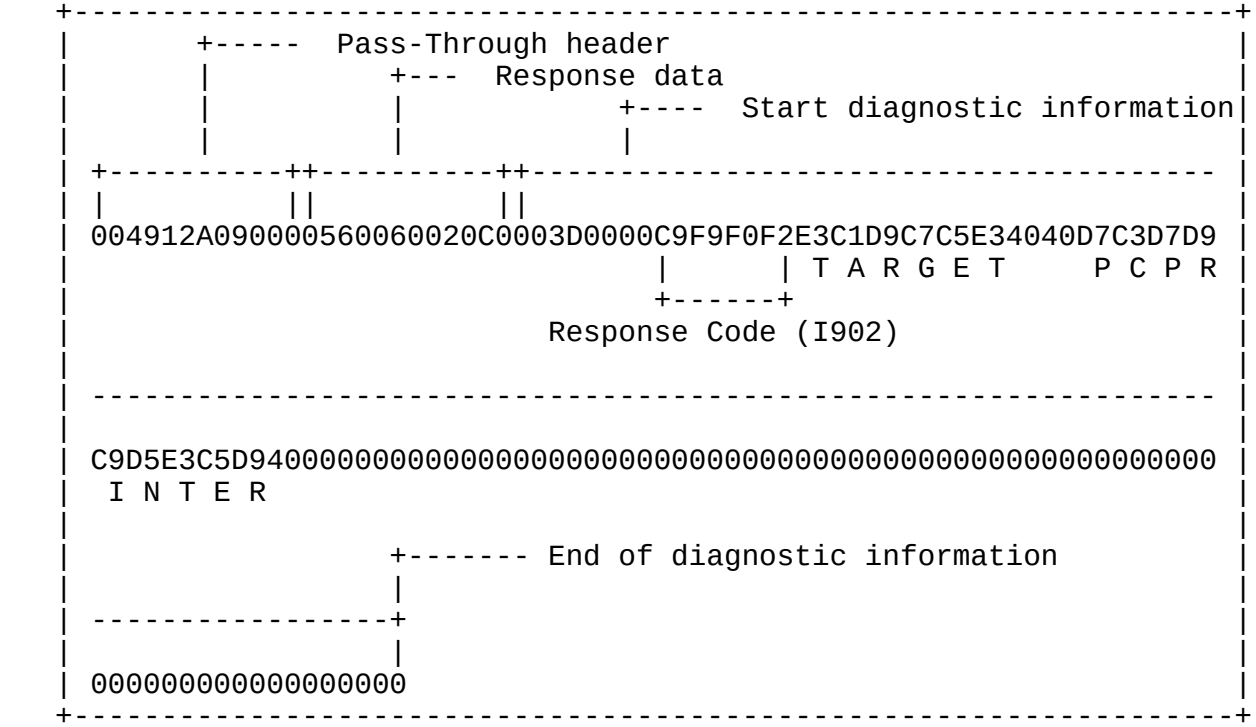


Figure 1. Example of a success response record.

- ’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)

9.2 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.

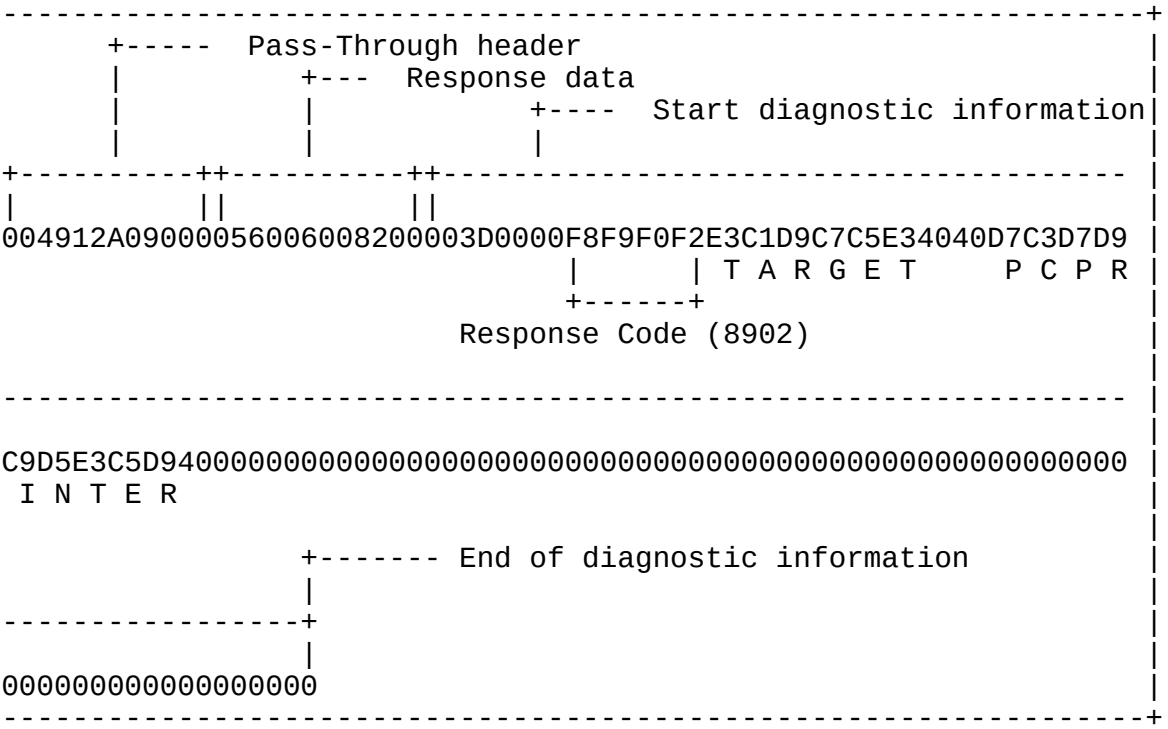


Figure 2. Example of an error response record.

- ’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)

9.3 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.

10. 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.

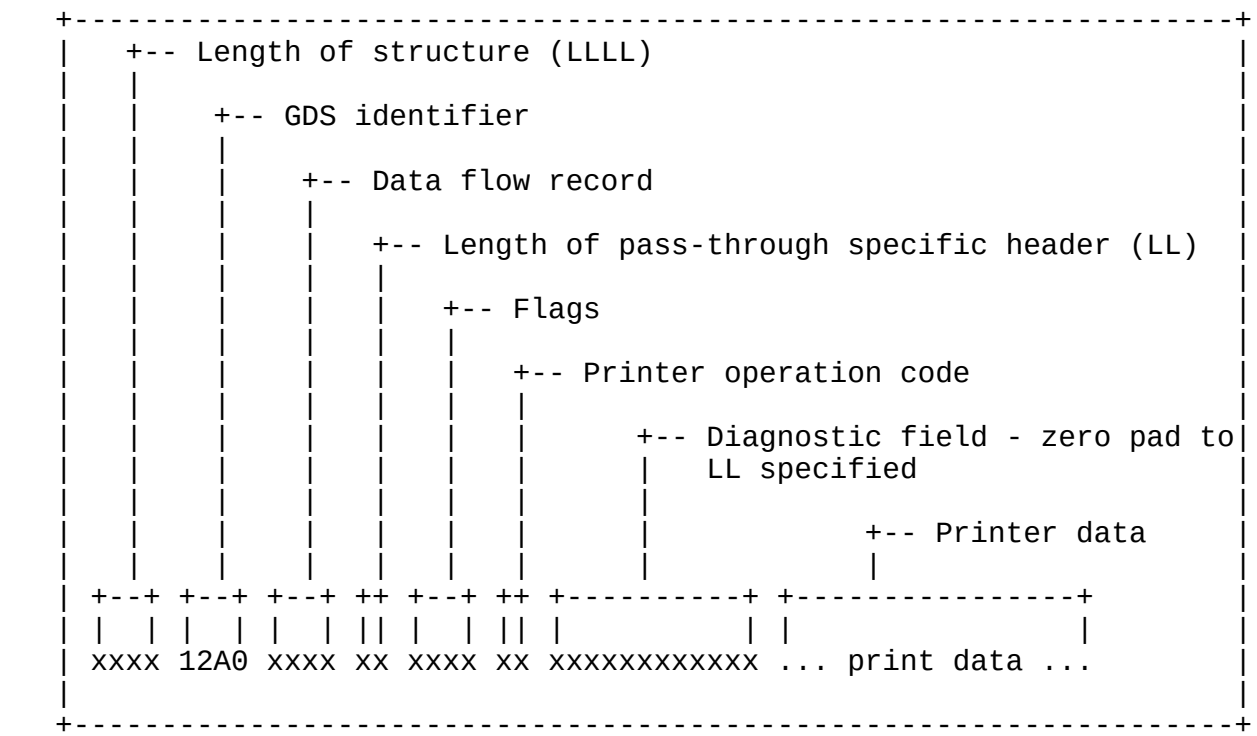


Figure 3. Layout of the printer pass-through header

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 = xx1x xxxx then bytes 10-LL may contain:

Printer ready	C9 00 00 00 02

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

Command/parameter not valid	C9 00 03 02 2x
Print check	C9 00 03 02 3x
Forms check	C9 00 03 02 4x
Normal periodic condition	C9 00 03 02 5x
Data stream error	C9 00 03 02 6x
Machine/print/ribbon check	C9 00 03 02 8x

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

Cancel	08 11 02 00
Invalid print parameter	08 11 02 29

Invalid print command	08 11 02 28

Diagnostic information 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 AS/400 printer documentation for more specific information on these data stream exceptions. Some 3812 and 5553 errors that may be seen:

Machine check	C9 00 03 02 11	
Graphics check	C9 00 03 02 26	
Print check	C9 00 03 02 31	
Form jam	C9 00 03 02 41	
Paper jam	C9 00 03 02 47	
End of forms	C9 00 03 02 50	
Printer not ready	C9 00 03 02 51	
Data stream - class 1	C9 00 03 02 66	loss of text
Data stream - class 2	C9 00 03 02 67	text appearance
Data stream - class 3	C9 00 03 02 68	multibyte control error
Data stream - class 4	C9 00 03 02 69	multibyte control parm
Cover unexpectedly open	C9 00 03 02 81	
Machine check	C9 00 03 02 86	
Machine check	C9 00 03 02 87	
Ribbon check	C9 00 03 02 88	

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.

10.1 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.

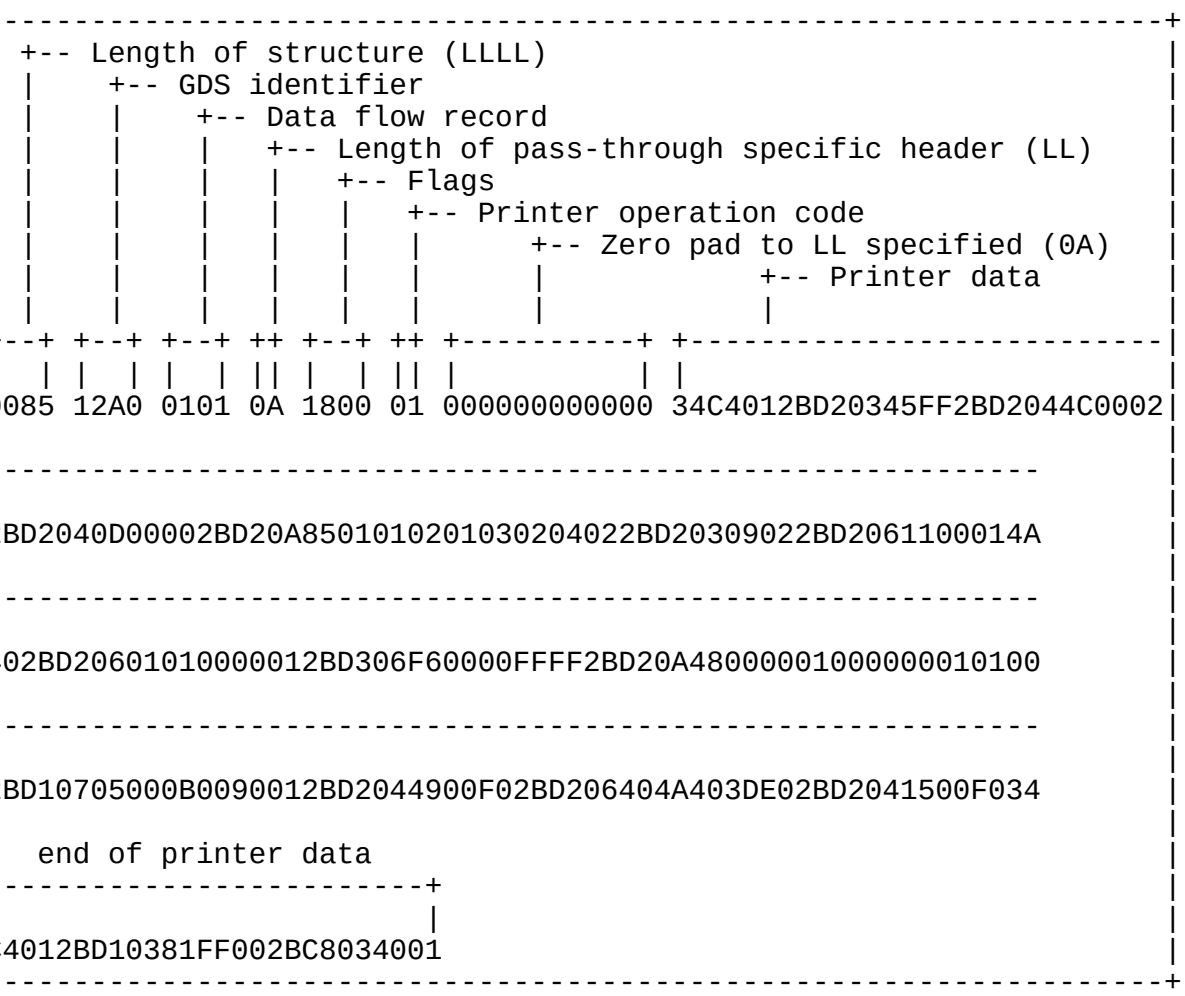


Figure 4. Server sending client data with a print record

- '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

10.2 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.

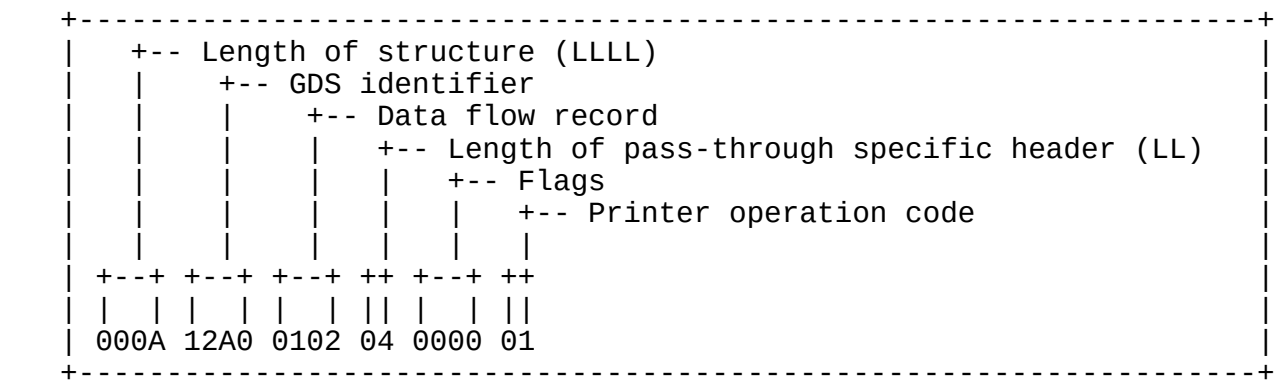


Figure 5. Client sending server a print complete record

- '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

10.3 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. The null data byte '00'X is optional, and in some cases may be omitted (in particular, this scenario occurs in DBCS print streams).

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.

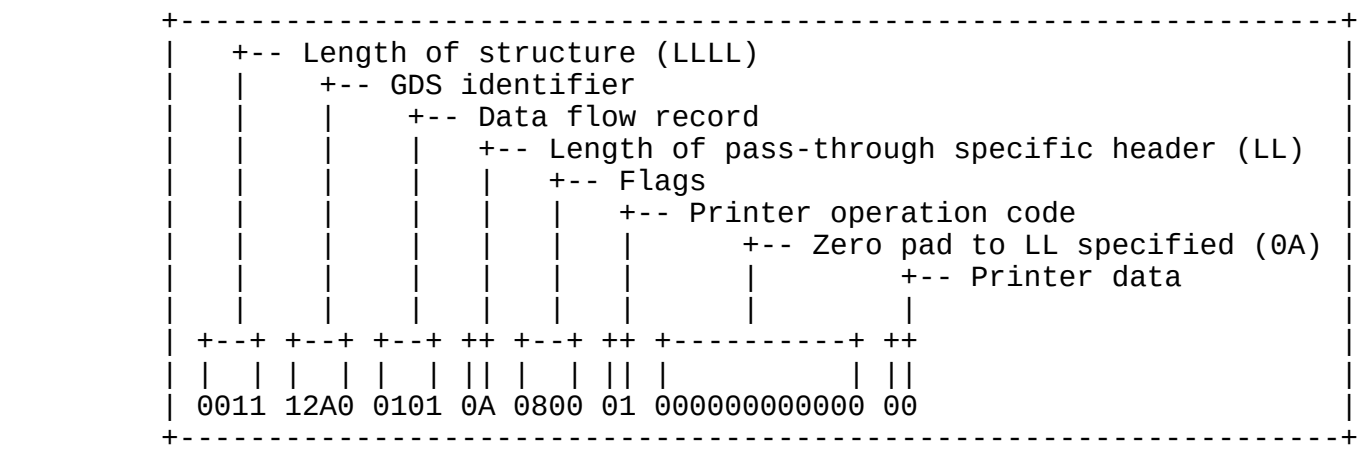


Figure 6. Server sending client a null print record

- '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

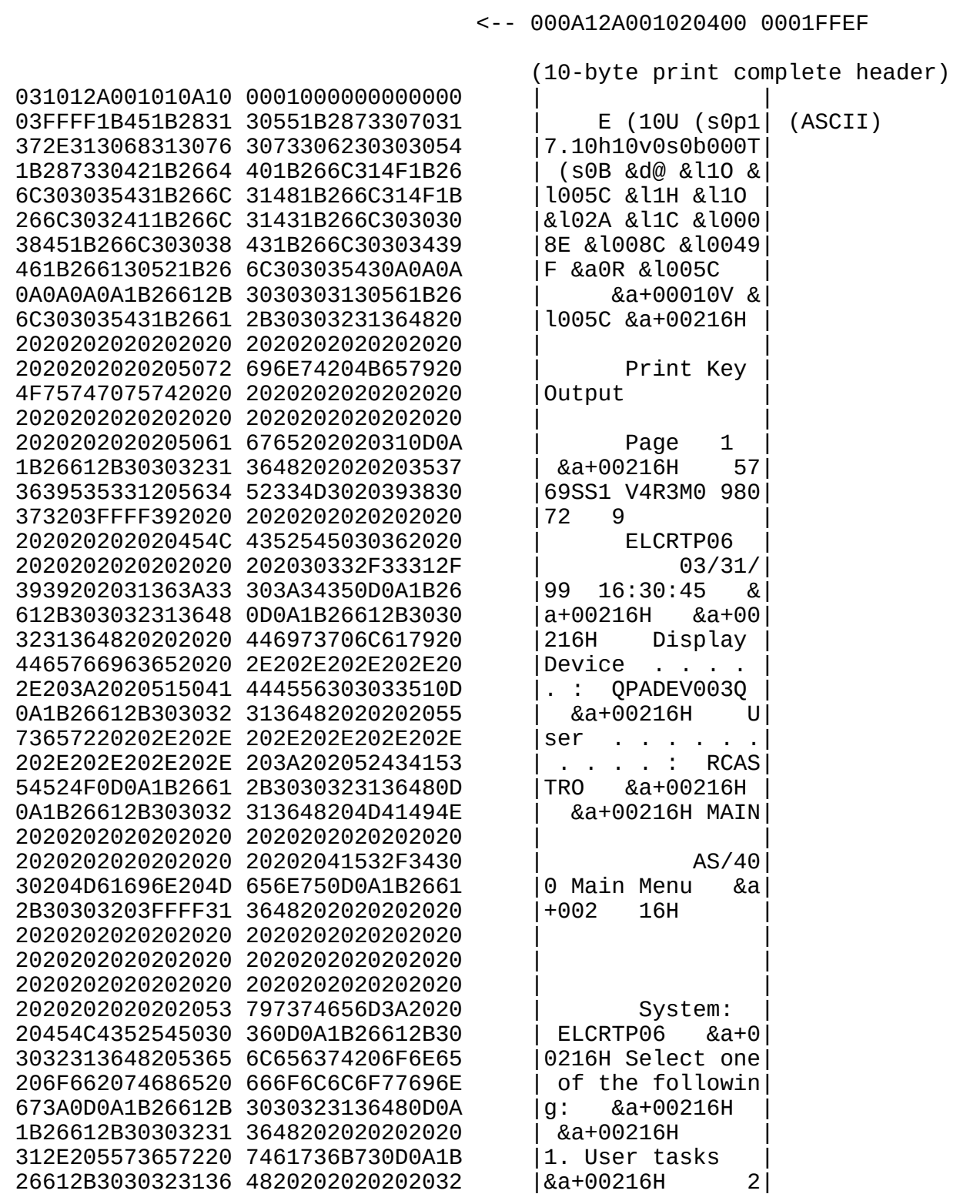
11. End-to-End Print Example

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
7EA5DFDDFD300404 0003FFF0       -->

(IAC DO TERMINAL-TYPE
IAC SB NEW-ENVIRON SEND USERVAR
```

```
IBMRSEED xxxxxxxx VAR USERVAR
IAC SE)

                                <-- FFFB18

                                    (IAC WILL TERMINAL-TYPE)

FFFA1801FFF0                    -->

(IAC SB TERMINAL-TYPE SEND IAC
 SE)

                                    FFFA27000349424D 52534545447EA5DF
                                    DDFD300404000344 45564E414D450144
                                    554D4D5950525403 49424D4D5347514E
                                    414D450151535953 4F50520349424D4D
                                    5347514C4942012A 4C49424C0349424D
                                    464F4E5401313103 49424D5452414E53
                                    464F524D01310349 424D4D4652545950
                                    4D444C012A485049 490349424D505052
                                    5352433101020103 49424D5050525352
                                    433201040349424D 454E56454C4F5045
                                    01FFFF0349424D41 5343494938393901
                                <-- 30FFF0

                                    (IAC SB NEW-ENVIRON IS USERVAR
                                     IBMRSEED xxxxxxxx VAR
                                     USERVAR DEVNAME VALUE DUMMYPRT
                                     USERVAR IBMMSGQNAME VALUE QSYSOPR
                                     USERVAR IBMMSGQLIB VALUE *LIBL
                                     USERVAR IBMFONT VALUE 11
                                     USERVAR IBMTRANSFORM VALUE 1
                                     USERVAR IBMMFRTYPMDL VALUE *HPII
                                     USERVAR IBMPPRSRC1 VALUE ESC '01'X
                                     USERVAR IBMPPRSRC2 VALUE '04'X
                                     USERVAR IBMENVELOPE VALUE IAC
                                     USERVAR IBMASCII899 VALUE 0
                                     IAC SE)

                                <-- FFFA180049424D2D 333831322D31FFF0

                                    (IAC SB TERMINAL-TYPE IS
                                     IBM-3812-1 IAC SE)
FFFD19                          -->

(IAC DO EOR)
                                <-- FFFB19
```

```
                                    (IAC WILL EOR)

 FFFB19                         -->

 (IAC WILL EOR)
                                <-- FFFD19

                                    (IAC DO EOR)
 FFFD00                         -->

 (IAC DO BINARY)
                                <-- FFFB00

                                    (IAC WILL BINARY)
 FFFB00                         -->

 (IAC WILL BINARY)
                                <-- FFFD00

                                    (IAC DO BINARY)

 004912A090000560 060020C0003D0000     |       -   {    |
 C9F9F0F2C5D3C3D9 E3D7F0F6C4E4D4D4     |I902ELCRTP06DUMM| (EBCDIC)
 E8D7D9E340400000 0000000000000000     |YPRT            |
 0000000000000000 0000000000000000     |                |
 0000000000000000 00FFEF           --> |                |

 (73-byte startup success response
  record ... IAC EOR)
 00DF12A001010A18 0001000000000000     |                |
 03CD1B451B283130 551B287330703130     |   E (10U (s0p10| (ASCII)
 2E30306831327630 733062303033541B     |.00h12v0s0b003T |
 287330421B266440 1B266C304F1B266C     |(s0B &d@ &l0O &l|
 303038431B266C30 3035431B28733070     |008C &l005C (s0p|
 31372E3130683130 7630733062303030     |17.10h10v0s0b000|
 541B283130551B28 73307031372E3130     |T (10U (s0p17.10|
 6831307630733062 303030541B287330     |h10v0s0b000T (s0|
 421B2664401B266C 314F1B266C303035     |B &d@ &l1O &l005|
 431B287330703137 2E31306831307630     |C (s0p17.10h10v0|
 733062303030541B 266C314F1B287330     |s0b000T &l1O (s0|
 7031372E31306831 3076307330623030     |p17.10h10v0s0b00|
 30541B2873307031 372E313068313076     |0T (s0p17.10h10v|
 3073306230303054 1B266C30303543FF     |0s0b000T &l005C |
 EF                                --> |                |

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


<-- 000A12A001020400 0001FFEF
(10-byte print complete header)
031012A001010A10 0001000000000000
03FFFF1B451B2831 30551B2873307031
372E313068313076 3073306230303054
1B287330421B2664 401B266C314F1B26
6C303035431B266C 31481B266C314F1B
266C3032411B266C 31431B266C303030
38451B266C303038 431B266C30303439
461B266130521B26 6C303035430A0A0A
0A0A0A0A1B26612B 3030303130561B26
6C303035431B2661 2B30303231364820
2020202020202020 2020202020202020
2020202020205072 696E74204B657920
4F75747075742020 2020202020202020
2020202020202020 2020202020202020
2020202020205061 6765202020310D0A
1B26612B30303231 3648202020203537
3639535331205634 52334D3020393830
373203FFFF392020 2020202020202020
202020202020454C 4352545030362020
2020202020202020 202030332F33312F
3939202031363A33 303A34350D0A1B26
612B303032313648 0D0A1B26612B3030
3231364820202020 446973706C617920
4465766963652020 2E202E202E202E20
2E203A2020515041 444556303033510D
0A1B26612B303032 3136482020202055
73657220202E202E 202E202E202E202E
202E202E202E202E 203A202052434153
54524F0D0A1B2661 2B30303231364800D
0A1B26612B303032 313648204D41494E
2020202020202020 2020202020202020
2020202020202020 20202041532F3430
30204D61696E204D 656E750D0A1B2661
2B30303203FFFF31 3648202020202020
2020202020202020 2020202020202020
2020202020202020 2020202020202020
2020202020202020 2020202020202020
2020202020202053 797374656D3A2020
20454C4352545030 360D0A1B26612B30
3032313648205365 6C656374206F6E65
206F662074686520 666F6C6C6F77696E
673A0D0A1B26612B 3030323136480D0A
1B26612B30303231 3648202020202020
312E205573657220 7461736B730D0A1B
26612B3030323136 4820202020202032
| | |
| E (10U (s0p1| (ASCII)
|7.10h10v0s0b000T|
| (s0B &d@ &l1O &|
|l005C &l1H &l1O |
|&l02A &l1C &l000|
|8E &l008C &l0049|
|F &a0R &l005C |
| &a+00010V &|
|l005C &a+00216H |
| |
| Print Key |
|Output |
| |
| Page 1 |
| &a+00216H 57|
|69SS1 V4R3M0 980|
|72 9 |
| ELCRTP06 |
| 03/31/|
|99 16:30:45 &|
|a+00216H &a+00|
|216H Display |
|Device . . . . |
|. : QPADEV003Q |
| &a+00216H U|
|ser . . . . . .|
| . . . . : RCAS|
|TRO &a+00216H |
| &a+00216H MAIN|
| |
| AS/40|
|0 Main Menu &a|
|+002 16H |
| |
| |
| |
| System: |
| ELCRTP06 &a+0|
|0216H Select one|
| of the followin|
|g: &a+00216H |
| &a+00216H |
|1. User tasks |
|&a+00216H 2|

```
2E204F6666696365 207461736B730D0A     |. Office tasks  |
1B26612B30303231 36480D0A1B26612B     | &a+00216H   &a+|
3030323136482020 20202020342E2046     |00216H      4. F|
696C65732C206C69 627261726965732C     |iles, libraries,|
20616EFFEF                            | an             |

(... 784-byte print record ...
 ... first of chain ... IAC EOR)
                                 <-- 000A12A001020400 0001FFEF

                                      (10-byte print complete header)

020312A001010A00 0001000000000000     |                |
64206603FFFF6F6C 646572730D0A1B26     |d f   olders   &| (ASCII)
612B303032313648 0D0A1B26612B3030     |a+00216H   &a+00|
3231364820202020 2020362E20436F6D     |216H      6. Com|
6D756E6963617469 6F6E730D0A1B2661     |munications   &a|
2B3030323136480D 0A1B26612B303032     |+00216H   &a+002|
3136482020202020 20382E2050726F62     |16H      8. Prob|
6C656D2068616E64 6C696E670D0A1B26     |lem handling   &|
612B303032313648 202020202020392E     |a+00216H      9.|
20446973706C6179 2061206D656E750D     | Display a menu |
0A1B26612B303032 3136482020202020     |  &a+00216H     |
31302E20496E666F 726D6174696F6E20     |10. Information |
417373697374616E 74206F7074696F6E     |Assistant option|
730D0A1B26612B30 3032313648202020     |s   &a+00216H   |
202031312E20436C 69656E7420416363     |  11. Client Acc|
6573732F34303020 7461736B730D0A1B     |ess/400 tasks   |
26612B3030323136 480D0A1B26612B30     |&a+00216H   &a+0|
303231364803ED20 2020202039302E20     |0216H       90. |
5369676E206F6666 0D0A1B26612B3030     |Sign off   &a+00|
323136480D0A1B26 612B303032313648     |216H   &a+00216H|
2053656C65637469 6F6E206F7220636F     | Selection or co|
6D6D616E640D0A1B 26612B3030323136     |mmand   &a+00216|
48203D3D3D3E0D0A 1B26612B30303231     |H ===>   &a+0021|
36480D0A1B26612B 3030323136482046     |6H   &a+00216H F|
333D457869742020 2046343D50726F6D     |3=Exit   F4=Prom|
707420202046393D 5265747269657665     |pt   F9=Retrieve|
2020204631323D43 616E63656C202020     |   F12=Cancel   |
4631333D496E666F 726D6174696F6E20     |F13=Information |
417373697374616E 740D0A1B26612B30     |Assistant   &a+0|
3032313648204632 333D53657420696E     |0216H F23=Set in|
697469616C206D65 6E750D0A1B26612B     |itial menu   &a+|
3030323136480D0A 1B26612B30303231     |00216H   &a+0021|
36480D0CFFEF                          |6H              |

(... 515-byte print record ...
 IAC EOR)
```

```
                                  <-- 000A12A001020400 0001FFEF

                                      (10-byte print complete header)
 001412A001010A00 0001000000000000    |                |
 03021B45FFEF                         |   E            |  (ASCII)

 (... 20-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)
```

12. Authors' Note

Discussion of this memo 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.

13. References

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

[2] IBM, "5494 Remote Control Unit, Functions Reference", SC30-3533-04, August 1995.

[3] IBM, "AS/400 System API Reference", SC41-5801-01, February 1998.

[4] IBM, "AS/400 TCP/IP Configuration and Reference", SC41-5420-02, September 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", STD 8, RFC 854, May 1983.

[9] Postel, J. and J. Reynolds, "Telnet Option Specifications", STD 8, RFC 855, May 1983.

[10] Postel, J. and J. Reynolds, "Telnet Binary Transmission", STD 27, RFC 856, May 1983.

[11] VanBokkeln, J., "Telnet Terminal-Type Option", RFC 1091, February 1989.

[12] Postel, J. and J. Reynolds, "Telnet End of Record Option", RFC 885, December 1983.

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

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

[15] Postel, J. and J. Reynolds, "Telnet Supress Go Ahead Option", STD 29, RFC 858, May 1983.

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

[17] Data Encryption Standard (DES), Federal Information Processing Standards Publication 46-2, January 22, 1988.

[18] DES Modes of Operation, Federal Information Processing Standards Publication 81, December 1980.

[19] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[20] IBM, "IBM Pageprinter 3812 Programming Reference", S544-3268.

14. Security Considerations

Security considerations of passwords are discussed in Section 6.

15. Authors' Addresses

Thomas E. Murphy, Jr.
IBM Corporation
1701 North Street
Endicott, NY 13760

Phone: (607) 752-5482
Fax: (607) 752-5421
EMail: murphyte@us.ibm.com

Paul F. Rieth
IBM Corporation
1701 North Street
Endicott, NY 13760

Phone: (607) 752-5474
Fax: (607) 752-5421
EMail: rieth@us.ibm.com

Jeffrey S. Stevens
IBM Corporation
1701 North Street
Endicott, NY 13760

Phone: (607) 752-5488
Fax: (607) 752-5421
EMail: jssteven@us.ibm.com

16. Relation to Other RFC's

UPDATES

This memo 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 memo 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 [19].

17. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.