TN3270E Working Group G. Pullen
Internet-Draft: <draft-ietf-tn3270e-extensions-01.txt> M. Williams
Extends: RFC 2355 OpenConnect Systems
Expiration Date: November 2000 May 3, 2000

TN3270E Functional Extensions

Status of this Memo

Copyright Notice

Abstract

This draft addresses issues and implementation problems defined and discussed at the TN3270E/TN5250E Interoperability Events. It does not replace the current TN3270 Enhancements protocol. It describes functional extensions to the TN3270E protocol. The TN3270E function negotiation mechanism is used to allow the server and client to determine which, if any, of these functions will be supported during a session. This preserves backward compatibility between clients and servers that do not support these features.

Among the issues to be address by this draft are SNA/TN3270E Contention state resolution, SNA Sense Code support, Function Management Header support, and TN3270E header byte-doubling suppression.

1. Table of Contents

2. Negotiated Function Codes

To maintain backward compatibility with clients and servers that do not support the extended TN3270E functions all new functionality will be negotiated. The current TN3270E function negotiation rules apply. Either side may request one or more of the extended functions by adding them to the function code list during TN3270E function negotiations. Either side may reject the function by removing it from the function list.

The extended TN3270E function negotiation codes are defined as:

CONTENTION-RESOLUTION	5
FMH-SUPPORT	6
SNA-SENSE	7
SUPPRESS-HEADER-BYTE-DOUBLING	8

3. Negotiated Function Example

The SNA-SENSE function support is enabled by the negotiation below:

```
Server:   IAC DO TN3270E
Client:   IAC WILL TN3270E
            . . .
Client:   IAC SB TN3270E FUNCTIONS REQUEST ... SNA-SENSE IAC SE
Server:   IAC SB TN3270E FUNCTIONS IS ... SNA-SENSE IAC SE
```

Support is disabled by the negotiation below (the server does not support the SNA-SENSE function):

```
Server:   IAC DO TN3270E
Client:   IAC WILL TN3270E
             . . .
Client:   IAC SB TN3270E FUNCTIONS REQUEST ... SNA-SENSE IAC SE
Server:   IAC SB TN3270E FUNCTIONS REQUEST ... IAC SE
Client:   IAC SB TN3270E FUNCTIONS IS ... IAC SE
```

4. Contention Resolution Function

This function addresses shortcomings in the current TN3270E (RFC 2355) specification that stem from the fact that SNA is a send/receive state oriented protocol, while TN3270E is relatively state free. The following subsections define the problems to be addressed and the methods to resolve those issues.

4.1 Keyboard Restore Problem

The Keyboard Restore problem concerns uncertainty over when the client can send data to the TN3270E server. TN3270E provides for an End-Of-Record (EOR) mechanism, which allows the client to determine where the boundary is between 3270 data stream commands. The server sends EOR whenever it sends data to the client for which the LIC (Last-In-Chain) indicator was set. Clients have no choice but to interpret the presence of EOR as an indication that it is okay to go ahead and send data back to the host (providing the 3270 data-stream has restored the keyboard). Since it is not uncommon for the Server to receive a LIC from the host with no CDI (Change Direction Indicator) set, a serious problem is created where the client will send data to the server when it does not own the send state.

The solution is for the server to provide the client with an indication that it may send data. The Send Data Indicator (SDI) mechanism will be discussed later in this document.

4.2 Implied Keyboard Restore Problem

The Implied Keyboard Restore problem occurs when an application never explicitly sets the keyboard restore bit of the WCC byte in

any of the 3270 data streams during a bracket. In SNA, EB is considered an "implied" keyboard restore in this case. However, since TN clients are not aware of the bracket or direction state the client is not aware that it is allowed to send data and often hangs in X-CLOCK state.

The solution for this problem is for the server to detect the implied keyboard restore condition and send the Keyboard Restore Indicator (KRI) flag to inform the client that its keyboard is unlocked (ready state).

4.3 BID Problem

In SNA, when a session is in the BETB (Between Bracket), the Primary LU (PLU), or host, may bid for the bracket by either sending an explicit or implicit BID. The Secondary LU (SLU), or terminal, processes the BID, either granting the bracket to the host or rejecting the request. Having granted the bracket the SLU must

enter the X-CLOCK (Time) input inhibited state.

An implicit BID occurs when the session is BETB and the host sends a message to the SLU with Begin Bracket (BB) indicated. No BID actually flows but is implied. The SLU may accept or reject as if a BID had been sent.

In the TN3270 world, there is no mechanism for including the client in the BID process. The server must process the BID on the client's behalf, without the ability to request the client yield the send state. This leads to a variety of problems when the client attempts to send data inbound after the server has sent positive response to a BID from the host. These problems include hung or lost sessions, lost data, or SNA or host application error messages, depending on data flow, timing, and how the server handles the BID process.

This problem can be addressed by allowing the BID to propagate to the client. When the server receives a valid BID (implicit or explicit) from the host (i.e. one that occurs in the BETB state) it will forward it to the client. The client will respond either positively or negatively. Having granted the BID (positive response), the client enters the X-CLOCK input inhibited state until the session reenters contention state.

4.4 Signal Problem

The Signal problem occurs when the PLU sends a Signal in order to force the SLU to yield direction. For example, when the secondary has rejected a BID and the host needs to override it. The BID reject may occur when the user types some data (perhaps an unintentional depression of the space bar) and does not press an AID key. The SNA architecture provides that the primary (host) can send a Signal. The secondary should reply with a positive response, send a null RU with Change Direction to yield direction (and Begin Bracket if appropriate), and enter send inhibit state.

With TN3270 there is no way for the server to force the client to yield the send state.

4.5 CONTENTION-RESOLUTION Implementation

This section defines a new negotiated TN3270E function called CONTENTION-RESOLUTION. Support of this function implies that both the client and the server are able to handle the SDI, KRI and Signal header flags and the BID data type as defined in this specification.

This function is intended SNA TN3270E environments only. Non-SNA server implementations should ALWAYS disable this function during TN3270E function negotiations.

When the CONTENTION-RESOLUTION function is supported, the REQUEST-FLAG header field is interpreted as a bit mask, instead of a

byte value, to allow the field to be used for Send Data, Keyboard Restore and Signal indicators.

4.5.1 Send Data Indicator (SDI)

To use the Send Data Indicator the CONTENTION-RESOLUTION function must be supported by and agreed upon by both the server and client during TN3270E function negotiations. SDI is only valid for TN3270E terminals in PLU-SLU session (3270-DATA type). SDI is not used for SSCP-LU mode to avoid the overhead of the server having to BID to send asynchronous SSCP-LU-DATA records to the client.

SDI is meaningful only when sent by the server. It is sent in the REQUEST-FLAG field of the TN3270E header. The SDI bit mask is:

```
SEND-DATA-MASK  0x01
```

A bit value of 1 (true) indicates to the client that it holds the send state. A bit value of 0 (false) indicates the server (and host by extension) holds the send state.

In SNA LU-LU session, the server sends SDI when the host relinquishes send state with either the CDI or the EBI set in the SNA RU header.

It is valid for the server to send a null 3270-DATA message (TN3270E header and EOR, no data) to indicate the send state to the client. This allows the server and client to handle a NULL RU containing EBI or CDI received from the host.

The server ignores SDI in messages from the client and processes any data as usual depending on data type.

When SDI is received by the client and the current TN3270E message has been processed (upon receipt of EOR) the client may send data to the server. If RESPONSES have been negotiated, the client must send RESPONSES to the server regardless of the send state. Upon receipt of SDI, the client must send all pending RESPONSE messages before sending any keyboard input to the server.

SDI is not a replacement for the 3270 data stream WCC Keyboard Restore bit. The client must track the 3270 WCC Keyboard Restore flag, TN3270E Keyboard Restore Indicator (KRI) and SDI to determine whether or not it can start sending data to the server. If keyboard restore (WCC or KRI) is received, the keyboard input must still be buffered until the SDI is received.

The client may send an ATTN key (IAC IP) regardless of the keyboard State, including input inhibited state. ATTN causes the server to send a SIGNAL to the host requesting Change Direction. This may allow the user to recover from a direction state timing or synchronization problem (i.e. server neglected to send SDI). The client should avoid subsequent ATTN keys until it receives direction

from the host. The server may disregard successive ATTN keys while waiting for the first ATTN to be processed and direction is yield by the host.

The client may also send SYSREQ (if enabled by TN3270E function negotiation) to override the input inhibit state. This allows the user to switch to SSCP-LU session (possibly to logoff).

The RESET key is used to clear local terminal and X-SYSTEM error conditions. RESET purges all buffered (type-ahead) keystrokes, except when entered to remove terminal Insert mode. In this case, a second RESET is required to purge the type-ahead buffer. RESET does restore the keyboard allowing the user to begin typing buffered keystrokes. However, it does NOT clear the X-CLOCK condition or allow the client to override the send state and forward data to the server.

4.5.2 Keyboard Restore Indicator (KRI)

To use the Keyboard Restore Indicator the CONTENTION-RESOLUTION Function must be supported by and agreed upon by both the server and Client during TN3270E function negotiations. KRI is only valid for TN3270E terminals in PLU-SLU session (3270-DATA type mode).

KRI is meaningful only when sent by the server. KRI is sent in the REQUEST-FLAG field of the TN3270E header. The KRI bit mask is:

```
KEYBOARD-RESTORE-MASK  0x02
```

A bit value of 1 (true) indicates to the client that its keyboard has been restored. The client's X-CLOCK indicator is turned off, allowing the user to enter data. However, the client may not send data to the server until it has also received SDI from the server (which may be set in the same REQUEST-FLAG field).

Logically, the client treats KRI the same as it does the 3270 WCC Keyboard Restore bit. KRI is not a replacement for the 3270 data stream WCC Keyboard Restore bit. The client must still track both the KRI and 3270 WCC Keyboard Restore flag to determine the keyboard state. Normally, one or the other will be received. However, the client should not balk if both are received on a 3270-DATA message.

The server ignores KRI in messages from the client and processes any data as usual depending on data type.

The server sends KRI when it detects an "implied" keyboard restore during LU-LU session. The server must track whether the host application has explicitly set the keyboard restore bit of the WCC byte in any of the 3270 data streams during a bracket. If not, the server must set KRI in the TN3270E message header when EB is set in the SNA header.

It is valid for the server to send a null 3270-DATA message (TN3270E Header and EOR, no data) to indicate the KRI to the client. This

allows the server and client to handle a NULL RU containing EBI received from the host.

4.5.3 BID Data Type

To use the BID data type the CONTENTION-RESOLUTION function must be supported by and agreed upon by both the server and client during TN3270E function negotiations. The BID data type message is only valid on terminal sessions in 3270-DATA (LU-LU) mode. The BID data type is not valid during SSCP-LU mode, NVT mode, or on printer sessions.

The BID DATA-TYPE code is defined as:

Data-type Name	Code	Meaning
BID	0x09	The server indicates that the host has sent an implicit or explicit BID by sending this data type to the client.

The server sends the new TN3270E BID data type to the client upon receipt of either an implicit or an explicit BID from the host. The server must never send BID to the client when the host already has direction (holds send state).

To send the BID data type the server inserts the BID data type in the DATA-TYPE field of the TN3270E header, inserts a null (0x00) in the REQUEST-FLAG field, inserts ALWAYS-RESPONSE (0x02) in the RESPONSE-FLAG field and fills in an appropriate SEQ-NUMBER. The server should use the next number in the progression of sequence numbers. An End-of-Record (EOR) is appended immediately after the TN3270E header (there is no data portion for a BID message).

The BID data type must always receive a response from the client regardless of whether the RESPONSES function is supported on the session. The client's positive or negative response to a BID should be exactly the same as those defined in the TN3270 Enhancements RFC, unless the SNA Sense Code Function (defined in section 6) is used by the client to communicate a more specific code. The SEQ-NUMBER is returned by the client in its response, to allow the server to coordinate the response with the BID.

When the client receives a BID message it is accepted by returning a positive response, or rejected by returning a negative response. The format of a positive response is the same as the positive response defined for the TN3270E RESPONSES function (i.e., RESPONSE data type, POSITIVE-RESPONSE code in RESPONSE-FLAG field, SEQ-NUMBER from BID). When the client accepts the BID the keyboard state goes to input inhibited, the client displays the X-CLOCK symbol and may not send data until SDI is received from the server.

When the server receives a BID response from the client, it is

responsible for constructing the appropriate SNA response to the host.

If the client already has buffered data to be sent to the host the client can reject the BID. The negative response uses the TN3270E RESPONSES format (i.e., RESPONSE data type, NEGATIVE-RESPONSE code in RESPONSE-FLAG field, SEQ-NUMBER from BID). Unless the client supports the SNA Sense Codes function, there is no defined reason information in the data portion of the negative response. The server rejects the host's BID with a "Bracket Bid Reject" sense code (0x08130000). The client's send state should remain unchanged upon negatively responding to a BID (i.e. if send state is input inhibited, it stays that way).

If the client supports the SNA Sense Code function, it has the option of returning "Receiver in Transmit Mode" (0x081B0000) sense code. This may be returned to reject the Bid when the user has started typing data but has not yet pressed an AID key.

A potential race condition exists, where the client sends data at the same time the server is sending a BID to the client. The race condition is handled by the server, and is relatively transparent to the client. When the server receives data before the expected response to the BID, the data is treated as an implied negative response. The server sends the Bracket Bid Reject (0x08130000) negative response to the host's BID and forwards the client's data to the host. When the client's response to the BID is received it is discarded by the server. The client's keyboard state should be input inhibited, whether it responded positively or negatively to the BID, because it has not received SDI for the data it sent previously.

4.5.4 SIGNAL Indicator

To use the SIGNAL indicator the CONTENTION-RESOLUTION function must be supported by and agreed upon by both the server and client during TN3270E function negotiations. The SIGNAL indicator is only valid on BID data type messages. The SIGNAL indicator is sent in the REQUEST-FLAG field of the TN3270E header.

The SIGNAL bit mask is:

```
SIGNAL-MASK  0x04
```

A bit value of 1 (true) indicates that a Signal has been received from the PLU. Therefore the BID is "Forced" and the client MUST forfeit the send state.

The client must always respond to a BID with the SIGNAL indicator, as described in the BID section. It is not necessary for the client to echo the SIGNAL indicator in its response. However, the server should not balk if the client does echo the SIGNAL indicator. The

server must maintain in it's state machine that it is awaiting a response to a SIGNAL indicator.

When a Signal is received from the PLU the TN3270E Server's behavior may be summarized as follows:

```
Send positive response to the host for the Signal
If the host already has direction, or in contention state. . .
   there is nothing more to do
Else client has direction. . .
   send BID with Signal to client and wait for reply or data
   If data received first. . .
      forward data to host as normal (will carry CD)
   Else response received first. . .
      send null RU CD to Host (with BB if necessary)
```

Upon receipt of Signal from the host, the server returns positive response to the host, regardless of whether the host or client holds direction. If the host holds direction (send state), there is nothing more to be done. The client should already be awaiting data from the host.

If the client holds direction, the server sends a BID with the SIGNAL indicator set to inform the client that it no longer holds send state and its keyboard state is input inhibited. The server will receive either data or a positive response from the client.

The server forwards any inbound data from the client to the host, while awaiting response to the signal BID. The inbound data record will cause the direction (CD) state to return to the host. When the positive response is received from the client the server has nothing further to do.

If the server receives only a response from the client, the server sends a null RU with Change Direction (CD) to the host. The client MUST return positive response to the server. If the client sends negative response to a SIGNAL, even though it is not allowed to do so, the server treats it as a positive response and handles it accordingly.

The Client's behavior when a BID containing the Signal indicator is summarized as:

```
Receive BID with Signal indicator
If client has direction and buffered keystrokes with AID. . .
   send first AID buffer
Else host has direction (race condition) or no AID. . .
   the buffered keystrokes are left unchanged
Return positive response to Signal
Enter X-CLOCK input inhibited mode
Buffer any keystrokes/AID typed after the Signal
```

A Signal does not cause the client to purge any buffered keystrokes. If the client holds direction when the Signal is received, it may send one buffered AID message (if any) before sending positive response to the Signal. If the Host already had direction (race condition) or no AID key is buffered, the type-ahead buffer is retained, as is.

The client then accepts the BID, and enters input inhibit mode. No further buffered data may be forwarded to the host until direction is returned to the client.

The following diagram illustrates how the client should handle buffered keystrokes relative to BID/SIGNAL processing:

```
|<--- Data typed before BID --->|<--- Data typed after BID --->|
| is displayed on the screen.   | is NOT displayed on screen.  |
```

This allows the host application to do a Read Buffer, update the portion of the screen it wants to change, put the cursor back to the right place for the suspended input and restore the keyboard. The client then streams the buffered keystrokes into the screen image. Upon completion of these processes the screen image should be restored correctly.

5. Function Management Header (FMH) Support Function

Function Management Headers are not permitted in LU2 or LU3. Initially, they were not used in LU1 either. Consequently, no provision was made for them in TN3270 or TN3270E. Subsequently, support for FMHs over LU1 has been added to SNA based applications and devices. Requirements to support LU1 DBCS (Kanji etc.) and IPDS printer applications are driving this effort for TN3270E FMH support.

A de facto standard has arisen for handling Structured Field data stream FMHs in both TN3270 and TN3270E. This de facto standard is referred to below as "silent FMH support". Only FMH1 is supported and merely forwarded, in both directions, as data. The receiver must recognize that the FMH is present by inspecting the first few bytes of the Telnet record and determining that they do not look like valid SCS data. This is workable because FMH1 is a fixed 6-byte string, and it only occurs at the start of a record.

The TN3270E FMH support function expands on silent FMH support by adding a mechanism for transferring FMHs through a server using a new TN3270E FMH data type. The main argument for adding this function is to allow clients and servers to formally determine whether the other side can "really" support FMH flows. Since, this functionality is negotiable, client/server vendors can make the determination of the merits of knowing whether the other side truly supports LU1 Function Management Headers.

5.1 FMH Overview

FMH usage in its simplest terms:

- There is only one FMH per chain, starting at the beginning of the chain.
- The FMH may be spread over multiple RUs if too long for one. The Format Indicator (FI) is 1 in the BC RU and 0 in the rest.

At the next level of complexity:

- There may be multiple FMHs, consecutively, at the start of the chain.
- The presence of an FMH following the current one is indicated by the concatenation flag at byte 1, bit 0 of the current.
- As with a single FMH, these FMHs may continue over several RUs, but only the first RU has the FI flag on. FI=0 on subsequent RUs.

The concatenation flag in the preceding FMH is sufficient to introduce it. However, the description of FMHs in [2] only requires a (concatenated sequence of) FMH to start at the start of an RU, not necessarily at the start of a chain. It states that any RU in the

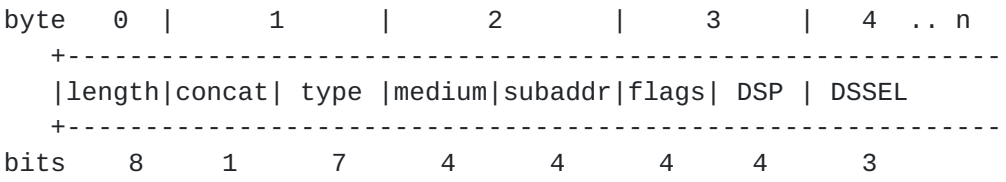
chain may have the FI flag on and thus start with an FMH, though the preceding RU ended with ordinary data.

This would be awkward for TN3270E, since the RU boundaries are not visible to the clients. Fortunately, it is awkward for higher-level Host applications also, e.g. CICS and IMS applications. These generally do not see RU boundaries either. Moreover, it is contradicted by the description of sense code 400F in [2]. So it is not surprising that the 3174 does not support this generality.

5.1.1 LU1 FMH1 Support

LU1 supports only FMH1. By default, LU1 sessions use SCS data stream. FMH1 is used to introduce support for an alternate data stream.

The FMH1 format is:



Where:

length	FMH length = (n+1)
concat	Concatenation Flag = (0)
type	FMH Type (FMH1 = 1)
medium	(console = 0)
subaddr	(0)
flags	Bit 0 - Send/Receive Indicator (SRI: Send=0, Receive=1)
DSP	Data-stream Profile - 0xB = Structured Field Data Stream - 0xD = IPDS Data Stream
DSSEL	Destination Selection

5.1.2 Usage of DSSEL in FMH1

FMH1 describes the data-stream of accompanying data. The accompanying data can be in a single chain (BEDS) or spread over multiple chains (BDS ... EDS). For TN3270E, the client is only able to support BEDS for inbound FMHs, because the server will assume CD at the end of each chain.

It is also possible to abort the data-stream (ADS instead of EDS), suspend a data-stream (SDS), and resume it later (RDS). In practice SDS/RDS are only used to insert console output into a longer transfer of data.

The entire data-stream must be within a bracket. If EB occurs after BDS but before EDS then the data-stream is implicitly aborted.

5.1.3 Structured Field Data Stream

This is used by DBCS and IPDS printers so that a Read Partition Query exchange can be conducted with an LU1 printer. (See [1].)

Outbound FMH1: 0x0601000B6000
Inbound FMH1: 0x0601008B6000 (SRI bit on)

BC RU
length = 6
concat = 0
DSP = 0xB
DSSEL = BEDS

Since the DSSEL = BEDS, the Structured Field data-stream is from the end of the FMH to the end of the chain.

The usage is bi-directional, but always as one from the host application and a reply from the secondary.

5.1.4 IPDS Data Stream

(See [4].)

Usage: 0x0601300D4000, 0x0601300D2000

OIC RU
length = 6
concat = 0
DSP = 0xD
DSSEL = BDS, EDS
No data following FMH in the same chain.

Although no ADS is sent, an EB before EDS implicitly aborts the data stream.

5.2 FMH Data Type

To use the FMH data type the FMH-SUPPORT function must be supported by and agreed upon by both the server and client during TN3270E function negotiations. The FMH data type message is only valid on LU1 printer sessions in SCS-DATA mode. The FMH data type is not valid during SSCP-LU mode, NVT mode, or on terminal or LU3 (DCS) printer sessions. The FMH DATA-TYPE code is defined as:

Data-type Name	Code	Meaning
FMH-DATA	0x0A	The sender indicates that the data portion of the message contains one or more Function Management Headers.

The FMH-DATA data type is bi-directional, meaning both the client and server can send this data type.

5.3 Server Implementation

If the FMH function has been negotiated, the server forwards the FMH data as part of the record, just as for normal data, and sets the FMH-DATA type in the TN3270E header.

If the FMH function was not negotiated the server may send the same with the SCS-DATA type. This maintains backward compatibility for servers that invoke silent FMH support.

There is a trade-off between making many data checks in the server, thereby keeping the client interface simple, and minimizing the server's knowledge of the data-stream, thereby preserving flexibility. This proposal takes the latter approach. In particular, except for silent FMH support, the server does not know which FMH types the client supports.

5.3.1 Bind Processing

Since TN3270E does not permit the client to reject the Bind, the server must police the bind parameters as far as possible.

If the server receives an LU1 Bind with byte 6 bit 1 set to 1 (FMHs will be used), but the client has not negotiated FMH function, then the server may choose to reject the Bind with sense 0x08350006. This is left optional (perhaps on customer configuration) in order to accommodate silent FMH support. However, when providing such support, the server is recommended to perform additional checks on the data, as outlined below.

5.3.2 Host/Server Flow

When the server receives an RU from the host application on an LU1 session FI = 1 and category = FMD, the server checks that the FMH is supported in principle. The server returns 0x400F0000 sense code if the Bind did not indicate FMHs or the FMH is not at the beginning of a chain. When providing silent FMH support to the client, the server may make the following optional checks, in this order:

Sense Code	Cause
0x10082009	Invalid header length (must be 6).
0x1008C000	Invalid FMH type (must be 1).
0x10086006	Invalid Data-stream Profile (must be 0xB or 0xD).
0x10080000	Other invalid parameters in FMH.

Example:

```
0x0601000B6000
0x0601300D4000
0x0601300D2000
```

The server either sends a negative response to the Host application or forwards the data to the client.

Note: If neither the FMH nor the SNA-SENSE functions are negotiated then it is recommended that the server only permit a specific list of FMHs from the Host.

The existing function is to send EOJ to the client. There is no change required here.

5.3.3 Client/Server Flow

When the server receives an FMH-DATA record from the client, it forwards the record to the Host application with FI set in the Begin Chain RU.

If the server receives a message FMH-DATA type but the FMH function was not negotiated, the server may choose either of two actions:

- Terminate the LU-LU session. It is suggested that, if BIND was negotiated, the UNBIND should carry a reason code of 0xFE. (If some future extension allows for SNA sense codes to flow to the client in the unbind image, the code to be used here should be 0x400F0000.)

- Behave, for that message only, as though FMH function was negotiated.

The server is not required to validate FMH-DATA messages received from the client.

If the server has sent a silent FMH (SCS-DATA type) to the client, the server must compare the first 6 bytes of the data for being 0x0601008B6000. If so, it sets FI in the Begin Chain RU.

5.3.4 FMH Responses

If SNA-SENSE-CODE is not supported, and the client returns a negative response to a silent FMH (SCS-DATA) or FMH-DATA type, the server is unable to determine whether the client objects to the FMH or the ensuing data. However, of the identified FMHs requiring support, only the Structured Field Data Stream can have data in the same chain. Even then, the cause of the rejection is likely to be that the client does not support FMHs. Therefore, it is recommended to the server interpret the negative response as a rejection of the FMH itself. The server should send negative response with 0x10080000 Sense code to the Host.

If SNA-SENSE-CODE is supported the server takes the first four bytes of data following the TN3270E header as an SNA sense code and sends these, unchanged and unchecked, in the negative response to the host. There are many SNA sense codes associated with FMH errors that the client may return. Most are at the application level and begin with

0x1008.

In addition to codes previously defined, below are some common FMH Sense codes:

Sense Code	Cause
0x10080000	Invalid parameters in FMH.
0x08350006	Bind has byte 6 bit 1 set to 1 (FMHs will be used) but printer does not support FMHs.
0x400F0000	Incorrect use of FI (not BC RU), or Bind error (byte 6/bit 1 set to 0). The FI flag is not echoed in the SNA response.

5.4 Client Implementation

A client that negotiates FMH function takes responsibility for validating the FMH-DATA messages. If an error is found in a received FMH, the client must send a NEGATIVE-RESPONSE.

If SNA-SENSE has been negotiated, the SNA-SENSE is set in the Response header field with the appropriate 4-byte SNA sense code in data field. Otherwise, the response field is set to NEGATIVE-RESPONSE and the data field contains the one byte Command Reject (0x0) status code.

A client that negotiates the FMH function must set FMH-DATA type on all records it sends that start with an FMH.

If a client receives EOJ after BDS and before EDS then the client should infer ADS.

6. SNA Sense Code Function

This function is intended for SNA TN3270E environments only. Non-SNA server implementations should ALWAYS disable this function during TN3270E function negotiations.

When the server and client operate in an SNA environment, it is impractical to perpetuate the one-byte error code mapping style of TN3270E. Especially, when SNA already provides a table of defined Sense codes. The SNA Sense Code function allows the client to return SNA Sense codes to the server, which are in turn forwarded to the SNA Host as a negative response.

The client indicates that the data portion of the response message contains a 4-byte SNA sense code by setting the following code in the RESPONSE-FLAG field:

SNA-SENSE-CODE 2

The SNA-SENSE function may be negotiated on either terminal or printer sessions. When the SNA-SENSE and RESPONSES functions have been negotiated, the server is committed to accepting SNA-SENSE-CODE responses to 3270-DATA, SCS-DATA (LU1), BID and FMH-DATA data type messages.

The client retains the option of providing specific SNA Sense codes, or letting the server map all errors to the appropriate SNA sense codes.

7. TN3270E Header Byte-doubling Suppression Function

A performance bottleneck facing Telnet server and client vendors is, any 0xFF within an outbound data stream must be byte-doubled (a second 0xFF inserted into the data stream) by the sender in order to differentiate actual data from Telnet IAC commands. The receiver of the data stream must then scan through the data stream removing the inserted 0xFF bytes. With header-based protocols, like TN3270E, Telnet byte-doubling forces the header to be variable length, to allow for any 0xFF bytes that may occur within the header.

From discussions on the TN3270E list, it was determined that Telnet Byte-doubling Suppression would best be handled outside of the TN3270E standard as a new Telnet negotiated option. This will allow other block mode protocols (i.e. traditional TN3270, and TN5250) to take advantage of the proposed option.

However, the variable length header issue is within the scope of the TN3270E standard. This draft proposes a method to make the TN3270E header fixed length by eliminating byte-doubling of the 5 header bytes. This function will extend the TN3270E standard to address this issue. Although this function is proposed in anticipation of a new Suppress Byte-doubling Telnet option, it is intended to be independent of whether such a Telnet option is negotiated.

When the SUPPRESS-HEADER-BYTE-DOUBLING function is enabled the TN3270E header will never be byte-doubled in either direction (client to server/server to client). Therefore, the size of the TN3270E header will ALWAYS be 5 bytes when the Header Byte-doubling function is in effect.

The sender of a TN3270E message guarantees that the first five (header) bytes of the record will not contain any embedded Telnet commands. The sender must byte-double the data portion of the TN3270E message.

The receiver of the message must validate that the message received does begin with a valid TN3270E header, to avoid misinterpreting asynchronous Telnet command packets between TN3270E records. The receiver must also be cognizant of whether a TN3270E header is expected to avoid problems that may occur if bytes in the middle of a chain of buffers are not scanned properly. When the receiver has determined that a valid TN3270E header is present it must skip past the header to begin scanning for byte-doubled 0xFF characters.

8. References

[1] IBM's "3174 Functional Description", Bookshelf book CN7A7003, GA23-0218-11.
[2] IBM's "Systems Network Architecture Formats", GA27-3136-14.
[3] RFC 2355
[4] IBM's "IPDS and SCS Technical Reference", S544-5312-00.

9. Term Definitions

This section defines some of the terms used in this document.

Input Inhibited -
a state where the client does not hold send state. Either the client has presented an AID message to the host or the host has gained direction via the BID process. The keyboard state is any of the type-ahead or keyboard disabled states.

Only SYSREQ or ATTN may be forwarded to the server while the client is in Input Inhibited state. SYSREQ and ATTN should never be buffered by the client.

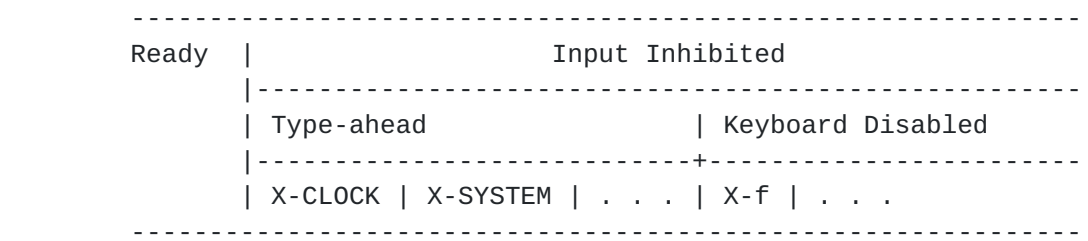
Keyboard Disabled -
a keyboard state where keystrokes may NOT be buffered by the client. Keyboard disabled states include (X-f), etc.

Keyboard States -
define the various modes a keyboard may be in during a TN3270E session.

When the client holds send state the keyboard is in Ready state. The client is free to process all keyboard input and forward any entered or buffered AID data to the server.

An Input Inhibited state is entered when the client surrenders send state by sending an AID buffer or granting a BID request from the host.

The diagram below summarizes the various keyboard states:



Type-ahead -
is a state in which the client (terminal) may buffer keystrokes when the keyboard is an Input Inhibited state. Displayed keyboard state may be either X-CLOCK (Time) or X-SYSTEM (System Lock). Keystrokes (text and AID keys) are buffered waiting for send state to be returned to the client.

10. Abbreviations

ADS	Abort Destination Selection, value of DSSEL
BB	Begin Bracket, byte 2 bit 0 of RH of BC RU
BC	Begin chain, byte 0 bit 6 of RH
BC RU	An RU with BC = 1
BDS	Begin Destination Selection, value of DSSEL
BEDS	Begin and End Destination Selection, value of DSSEL
DCA	Document Content Architecture
DSP	Data-stream Profile, byte 3 bits 4-7 of FMH
DSSEL	Destination Selection, byte 4 bits 0-2 of FMH
EB	End Bracket, byte 2 bit 1 of RH of BC RU
EC	End chain, byte 0 bit 7 of RH
EC RU	An RU with EC = 1
EDS	End Destination Selection, value of DSSEL
FI	Format Indicator, byte 0 bit 4 of RH
FIC	First In Chain - an RU with BC = 1 and EC = 0
FMD	Function Management Data (user data, not FMH)
FMH	Function Management Header, a SNA data header
IPDS	Intelligent Printer Data Stream
LIC	Last In Chain - an RU with BC = 0 and EC = 1
LU	Logical Unit
LUn	Logical Unit Type n, n = 0, 1, 2, etc.
MIC	Middle In Chain - an RU with BC = 0 and EC = 0
OIC	Only In Chain - an RU with BC = 1 and EC = 1
RDS	Resume Destination Selection, value of DSSEL
RH	Request Header, 3 byte header on SNA RU
RU	Request Unit, an SNA frame starting with an RH
SDS	Suspend Destination Selection, value of DSSEL
SNA	Systems Network Architecture

11. Conventions

- Byte order is big-endian, e.g. bit 0 is the most significant bit.

12. Author's Note

Portions of this document were drawn from the following sources:

- Contention Resolution proposal by Rodger Erickson (Wall Data).
- SNA Sense Code and Function Management Header Support proposal by Derek Bolton (Cisco Systems).
- TN3270E Byte-doubling Suppression proposal by Marty Williams (OpenConnect Systems).
- Discussions on the TN3270E list and at the TN3270E/TN5250E Interoperability Events, 1997-1998. Particularly contributions by Jim Mathewson II (IBM), Derek Bolton, Michael Boe, and Diane Henderson (Cisco Systems).

13. Author's Address

Gene Pullen	Marty Williams
gbp@openconnect.com	martyw@cisco.com
OpenConnect Systems	Cisco Systems
2711 LBJ Freeway	7025 Kit Creek Rd.
Dallas, TX 75234	Research Triangle Park, NC 27709-4987

Draft Expiration Date: November 2000