**Annex I**(informative) **Data transparency**

**Description**

Messages using this document formats ensure that certain fields contain arbitrary binary data, often called transparent data. Such data may contain 8-bit bytes that do not represent printable characters or may represent illegal codes in some communication schemes. Hardware devices connected using synchronous communication protocols or protocols designed to front-end for them should have no problem transmitting transparent data.

However, there are many low-end devices that communicate using older dial protocols. These protocols may use character-by-character parity checking, thus limiting data bytes to seven data bits. They may also restrict allowed data to printable characters and specific control codes. Even some older synchronous protocols, like BSC, place limits on the presence and use of specific control codes.

**Common limitations**

**Protocols that transmit data using seven data bits and odd parity**

This is a technique left over from the era of the mechanical teletype. Although modern modems support the technique as a format, the receiving end may not check the received characters for parity. If a parity error is detected, there is often no message level recovery initiated to retransmit the data in error. Thus, there is little benefit left to the technique.

Most service providers of transaction-processing services can set up low-speed (2,400 bit/s or lower) modems to operate with eight data bits and no parity as readily as with seven data bits and odd parity. Service providers are generally insensitive to data content, so the presence of undefined or control codes in the message stream will typically have no effect on the transmission of data. Most support Microcom Network Protocols (MNPs), which provide data error detection and correction (MNP 4) and data compression (MNP 5).

When the eighth data bit is present, there are two possible values for all character codes. When character data is represented, the parity bit should default to 0 (equivalent to space parity). It is recommended that data be sent in the format eight data bits, no parity. If MNP 4 and higher error detection and correction are supported by the communicating modems, it is recommended that it be enabled.

**Session level protocol**

The common protocols for a dial device connecting to a host transaction processing system use a terse command set of single byte control codes to establish and control the session. Some of the most common codes are described as follows. ASCII representations of control codes are used.

— SYN: Sync character. Two to five SYN characters are sent before and after each message to allow the receiver to get its timing lined up and to prevent loss of the first or last character. Most modern modems do not need this but it is useful if the case of a noisy analogue line.

— ENQ: initial poll, host to terminal. “Do you have a message for me?” Also used terminal-to-host to request retransmission of a response message following time-out. Many service providers “spoof” the initial host ENQ when a terminal first dials in. In this way, the initial message request is being received and buffered at the entry point into a large packet-switching network while the host is being alerted to the connection. When the host sets up the session to the service provider’s entry point and sends its ENQ command, the buffered message is sent to the host at high speed.

— ACK: acknowledge, message received correct. Used in both directions. Not used in Visa 1. Some protocols use ACK 0 and ACK 1 alternately.

— NACK: negative acknowledge, message received with errors. Used in both directions. Not used in Visa 1.

— STX: start of text, start of message.

— ETX: end of text, end of message. Used in both directions.

— ETB: end of block, more to follow. Used in some batch-oriented protocols.

— LRC: longitudinal redundancy check. A checksum is calculated on all data between the STX and ETX characters (including the ETX character). The result is an 8-bit character, which may contain any value from 0 to 255, including values, which violate parity. If a message fails the LRC check, the receiver may respond with a NACK (Visa 2) or may ignore the message (Visa 1).

— EOT: end of transmission, disconnect. Sometimes used as a general error reset. Usage varies with different protocols.

It is recommended that low-end terminals continue to use terse command and control codes. Specific session control protocol will be addressed separately.

**Protocols that limit data to printable characters and specific control codes**

For example, if a Visa data stream included an ETX character, the message would immediately terminate and the following character would be interpreted as the LRC checksum. The two basic approaches to circumventing this problem have been to use an encoding scheme that will allow data to be represented at some cost in overhead or to modify the protocol to accept transparent data.

**Character encoding schemes**

In the first category are several techniques. In BSC communications, the DLE character signals the processor that transparent data are to be processed. DLE STX begins transparent text and DLE ETX ends transparent text. The only character not thus supported is the DLE character itself. DLE is represented by the DLE DLE pair.

Another common technique is to use character representation of hexadecimal data. This effectively doubles the size of the data represented but data is easily interpreted and may be of arbitrary length. Other encoding schemes have been suggested, such as six for eight or seven for eight. Such representations are more space-efficient but are less intuitive to work with on processing. They also require fixed-length data blocks in order to come out with completely filled data bytes.

**Transparent capable protocols**

In the second category may be included conversion of devices to use synchronous communications and a message-level protocol variant. Internet protocols use a variety of HDLC (the TCP portion of TCP/IP) to transmit data on a bit-wise basis rather than a byte-wise basis. The data is broken into packets, each of which is individually checked for integrity at low layers of the protocol. At the message level, there is no overhead associated with data integrity checking.

A variant of message-level protocol called Length Declaration Format (LDF) was developed to address issues with interfacing dial terminals into X.25 PADs and some of the associated timing problems. The technique is simple, robust and of general use. Each message is preceded by a four-byte header containing a control character, a byte count for the following message and a flag indicating the type of checksum processing.

a) Without error checking: - <SOH>; <byte count>; <blank>; <STX>; text… <ETX>;

b) With 8-bit LRC checking: - <SOH>; <byte count>; ‘L’ <STX>; text… <ETX>; <LRC>;

c) With CRC 16 checking: - <SOH>; <byte count>; ‘C’ <STX>; text… <ETX>; <CRC 16>.

The message is assumed to be the length declared in the 16-bit byte count (maximum message length is therefore 65,535 bytes). Checking the message consists of determining that it matches the byte count in length and that the ETX character is the last character of the message. Note that an ETX character within the text does not end the text.

Messages may be formatted with or without checksum processing, depending on whether error- correcting modems are used. If no checksum processing is done, the checksum processing flag is blank. If 8-bit LRC processing is used, the checksum processing flag in the header is an “L” and a one-byte LRC follows the ETX character. The LRC is calculated on all data following the STX for the declared length and is compared to the value following the ETX character. If they match, the message is acknowledged with an ACK; if not, the message is negatively acknowledged with a NACK.

If Cyclical Redundancy Check (CRC) 16 processing is used, the checksum processing flag in the header is a “C” and a two-byte CRC 16 follows the ETX character. The CRC 16 is calculated on all data following the STX for the declared length and compared with the value following the ETX character. If they match, the message is acknowledged with an ACK; if not, the message is negatively acknowledged with a NACK.

**Recommendations**

To maximize the chance of error free transmissions, the following recommendations should be followed:

a) Use synchronous communications protocols, where possible, thereby removing data integrity overhead from message level processing.

b) If dial asynchronous communication is used, the message protocol should be based on LDF formats. The message length should have no dependency on message data content.

c) Using a character-based representation of binary data is not a desirable option because it makes message formats ambiguous. However, if neither a) nor b) is practical it can be used, but messages from dial terminals using such an implementation have to be translated, both inbound and outbound, to remove device dependencies before the message can be processed in a host receiving message from multiple terminal types.

**Transaction versus batch protocols**

Many common dial applications use different communication protocols for different functions. For example, if a POS needs authorization for a sale, the POS initiates the call and the conversation is run on a strict request/response basis. At the end of the day, the host may initiate the call, and the conversation is run using a batch-oriented protocol with no acknowledgement of individual blocks unless they are to be retransmitted.

Batch-oriented protocols which do little data acknowledgement are left over from the era when communications lines were much slower and turning the line around to send a response often took even longer than sending the message. With modern communication equipment, modems normally communicate in full duplex mode, so there is no line turn-around time. Timesaving from retransmitting only individual blocks in error versus full transmission should more than compensate for the overhead of handshaking each packet. [HDLC uses a technique of acknowledging the last correctly received packet number after receiving a group of packets. The sender retransmits all packets from the error point (if any) forward.]

There is no particular reason why data messages and packets have to match one for one and the convention of setting up data collection applications to process 80-byte card images is quite archaic. Blocking multiple messages within a packet and use of common data-compression techniques in transmission should allow efficient batch-oriented transmissions while maintaining consistent data integrity. Regardless of which partner initiates the call or what type of data is sent, there should be a positive handshake for each data packet or group of data packets sent.