

Network Working Group  
Request for Comments: 1967  
Category: Informational

K. Schneider  
ADTRAN, Inc.  
R. Friend  
Stac Technology  
August 1996

## PPP LZS-DCP Compression Protocol (LZS-DCP)

### Status of This Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links.

The PPP Compression Control Protocol [2] provides a method to negotiate and utilize compression protocols over PPP encapsulated links.

This document describes the use of the Stac LZS data compression algorithm for compressing PPP encapsulated packets, using a DCP header [6]. This protocol is an enhanced version of the non-DCP (Option 17) PPP Stac LZS compression protocol [5], and will be referred to as the LZS-DCP Compression Protocol.

### Table of Contents

1.	Introduction .....	2
1.1	Licensing .....	3
1.2	Specification of Requirements .....	3
1.3	Terminology .....	3
2.	LZS-DCP Packets .....	4
2.1	Example LZS-DCP Packets .....	5
2.2	Padding .....	6
2.3	Reliability and Sequencing .....	6
2.4	Data Expansion .....	6
2.5	Packet Format .....	7
2.5.1	PPP Protocol .....	7
2.5.2	DCP-Header .....	8
2.5.3	History Number .....	9
2.5.4	Sequence Number .....	9
2.5.5	Data .....	10
2.5.6	Longitudinal Check Byte .....	10

2.5.7	Compressed Data .....	11
3.	Sending Compressed Datagrams .....	11
3.1	Transmitter Process .....	11
3.2	Receiver Process .....	12
3.3	History Maintenance .....	13
3.4	Anti-Expansion Mechanism .....	14
3.5	History Resynchronization Mechanism .....	14
4.	Configuration Option Format .....	15
	SECURITY CONSIDERATIONS .....	16
	REFERENCES .....	17
	CHAIR'S ADDRESS .....	17
	AUTHORS' ADDRESSES .....	18

## 1. Introduction

Starting with a sliding window compression history, similar to LZ1 [3], Stac Electronics developed a compression algorithm identified as Stac LZS. A PPP Compression Protocol for this compression algorithm was developed and published [5]. That protocol was taken as a basis for data compression work done in TIA for DSU/CSUs. As a part of that standardization process, the concept of a portable Data Compression Protocol (DCP) was introduced [6]. The resulting (pending) TIA/EIA-655 standard uses this LZS-DCP protocol, which incorporates DCP into a PPP compression protocol for Stac LZS. A very similar protocol is currently out for ballot in the Frame Relay Forum. (It is identical except for the size of the history number field.)

This publication of the LZS-DCP compression protocol is in the interest of providing a common compression protocol for Stac-LZS, and to provide features that are not available with the LZS compression protocol [5]. Some of the differences between the LZS-DCP and LZS (compression type 17) protocols are as follows:

- 1) LZS-DCP provides an option which allows packets containing uncompressible data to be transferred without requiring the compression history to be cleared, potentially allowing a higher compression ratio. A bit is included in the DCP header to indicate whether the packet contains compressed or uncompressed data.
- 2) LZS-DCP uses reset request and acknowledgment bits in the DCP header that is included on each packet rather than using CCP's reset request and acknowledge packets, which may result in fewer discarded data packets during the REQ/ACK handshake.
- 3) LZS-DCP allows simultaneous use of both sequence numbers and the LCB for compression error detection.

The Stac LZS compression algorithm supports both single and multiple compression histories. A single compression history will require the minimum amount of memory to implement, but may not provide as much compression as a multiple history implementation.

Often, many streams of information are interleaved over the same physical link. Each virtual connection will transmit data that is independent of other virtual connections. Using multiple compression histories can improve the compression ratio of a communication link by associating separate compression histories with separate virtual links of communication.

### 1.1. Licensing

Source and object licenses are available on a non-discriminatory basis. Hardware implementations are also available. Contact Stac Electronics (hardware.sales@stac.com) for further information.

### 1.2. Specification of Requirements

In this document, several words are used to signify the requirements of the specification. These words are often capitalized.

- |          |   |
|----------|---|
| MUST     | This word, or the adjective "required", means that the definition is an absolute requirement of the specification.  |
| MUST NOT | This phrase means that the definition is an absolute prohibition of the specification.  |
| SHOULD   | This word, or the adjective "recommended", means that there may exist valid reasons in particular circumstances to ignore this item, but the full implications MUST be understood and carefully weighed before choosing a different course.             |
| MAY      | This word, or the adjective "optional", means that this item is one of an allowed set of alternatives. An implementation which does not include this option MUST be prepared to interoperate with another implementation which does include the option. |

### 1.3. Terminology

This document frequently uses the following terms:

- |          |  |
|----------|--|
| datagram | The unit of transmission in the network layer (such as IP). A datagram may be encapsulated in one or more packets passed to the data link layer. |
|----------|--|

frame        The unit of transmission at the data link layer. A frame may include a header and/or a trailer, along with some number of units of data.

packet       The basic unit of encapsulation, which is passed across the interface between the network layer and the data link layer. A packet is usually mapped to a frame; the exceptions are when data link layer fragmentation is being performed, or when multiple packets are incorporated into a single frame.

peer         The other end of the point-to-point link.

silently discard

This means the implementation discards the packet without further processing. The implementation SHOULD provide the capability of logging the error, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.

## 2. LZS-DCP Packets

Before any LZS-DCP packets are communicated, PPP MUST reach the Network-Layer Protocol phase, and the CCP Control Protocol MUST reach the Opened state.

Exactly one LZS-DCP datagram is encapsulated in the PPP Information field, where the PPP Protocol field indicates type hex 00FD (compressed datagram) or type hex 00FB (Individual link compressed datagram). Type hex 00FD is used when compression is negotiated over a single physical link or when compression is negotiated over a single bundle consisting of multiple physical links. Type hex 00FB is used when compression is negotiated separately over individual physical links to the same destination. For more information, please refer to PPP Compression Control Protocol.

The maximum length of the LZS-DCP datagram transmitted over a PPP link is the same as the maximum length of the Information field of a PPP encapsulated packet.

Prior to compression, the uncompressed data begins with the PPP Protocol ID Field. Protocol-Field-Compression MAY be used on this value, if has been successfully negotiated for the link.

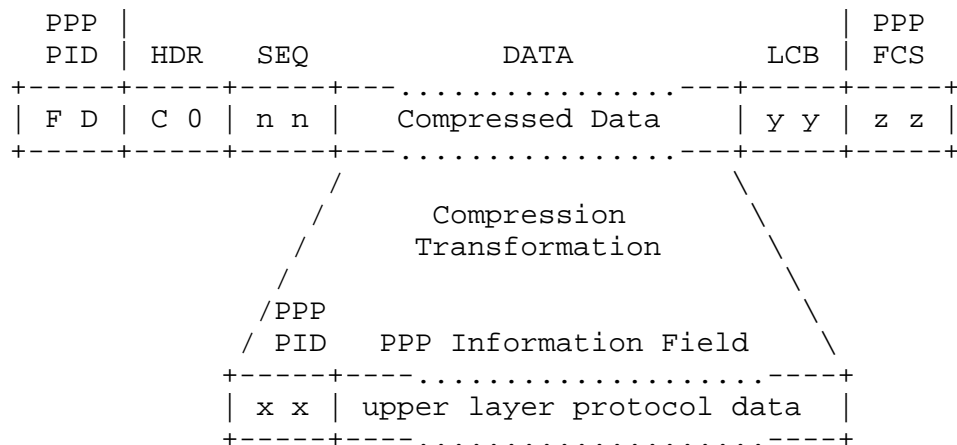
The PPP Protocol ID Field is followed by the original Information field. The length of the uncompressed data field is limited only by the allowed size of the compressed data field and the higher protocol

layers.

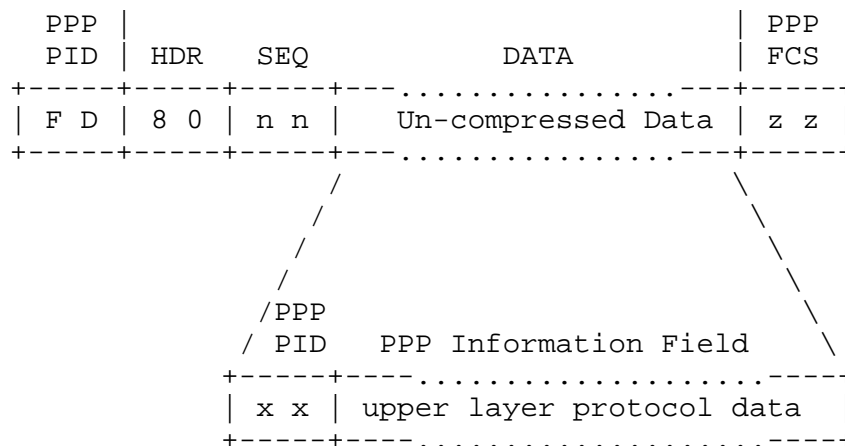
PPP Link Control Protocol packets MUST NOT be sent within LZS-DCP packets. PPP Network Control Protocol packets MUST NOT be sent within LZS-DCP packets.

- 2.1. Example LZS-DCP packets (shown using PPP in HDLC-like framing, using Address-and-Control-Field-Compression and Protocol-Field-Compression. - RFC 1662 )

Compressed Packet:



Uncompressed Packet



where: C0 and 80 are representative LZS-DCP headers; nn, xx, yy, and zz are values determined by the packet's context.

## 2.2. Padding

PPP padding is not allowed in a LZS-DCP packet. However, on compressed packets, padding may be accomplished by extending the data field with zeros following the last compressed data octet (see Section 2.1.1). This is referred to as LZS Padding. The LCB, if present, MUST be the octet preceding the frame CRC.

## 2.3. Reliability and Sequencing

When no Compression History is kept, the algorithm does not depend on a reliable link, and does not require that packets be delivered in sequence. However, per packet compression results in a lower compression ratio than it could be on a stream.

Some reasons for clearing the history on a per packet basis include:

- The link has a high error rate.
- The resources of the transmitter or receiver limit the ability to maintain a compression history between packets.

When one or more compression Histories are negotiated, the packet sequence MUST be preserved within specific History Numbers. There is no sequence requirement between different History Numbers.

When using one or more compression histories, the implementation MUST rely on either a lower layer reliable link protocol (RFC 1663), use a technique to keep the compressor and decompressor histories in synchronization, or both. The LZS-DCP protocol provides the Request-Req and Request-Ack bits in the DCP header for this purpose. Since this synchronization is done on a per history basis, the history number fields are required to be the same size in both directions of the link. Any data contained in the packet is processed after the signaling bits are processed.

The transmitter MAY clear a Compression History at any time.

The transmitter MUST clear a history after a receiving a Reset-Request for a given History Number.

## 2.4. Data Expansion

The maximum expansion of Stac LZS is 12.5%.

A Maximum Receive Unit (MRU) MAY be negotiated that is 12.5% larger than the size of a normal packet. Then, packets can always be sent compressed regardless of expansion.

The transmitter MAY send an uncompressed LZS-DCP packet at any time, although the typical use of uncompressed LZS-DCP packets is as an anti-expansion mechanism.

When the expansion plus compression header exceeds the size of the peer's MRU for the link, the data MUST be sent as an uncompressed LZS-DCP packet.

An uncompressed LZS-DCP packet is transmitted according to the format shown in Section 2.1, with the C/U bit set to 0 (Uncompressed-Data). If the Configuration Option Field 'Process Mode', is set to a value of 1 (Process-Uncompressed), uncompressed LZS-DCP packets are processed by both the compressor and the decompressor, updating the histories of each. If the Process Mode Field is set to a value of 0 (None), and the compressor has modified its history before sending the uncompressed packet, the compressor history MUST be clear.

## 2.5. Packet Format

A summary of the LZS-DCP packet format is shown below. The fields are transmitted from left to right.

```

      0                               1                               2
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           PPP Protocol           |   DCP-Header   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   (History Number)   |   (Seq Num)   |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|           Data ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   (LCB)   |
+---+---+---+---+---+---+

```

### 2.5.1. PPP Protocol

The PPP Protocol field is described in the Point-to-Point Protocol Encapsulation [1].

When the LZS-DCP compression protocol is successfully negotiated by the PPP Compression Control Protocol [2], the value is 00FD or 00FB hex. This value MAY be compressed when Protocol-Field-Compression is negotiated.

### 2.5.2. DCP-Header

The DCP-Header is nominally one octet in length, but may be extended through the use of the extension bit.

The format of the DCP-Header is as follows:

0	1	2	3	4	5	6	7
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+
E	C/U	R-A	R-R	Res	Res	Res	C/D
+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+	+-----+

E - Extension Bit

The E bit is the extension bit. If set to 0, it indicates that another octet of the DCP-Header is present. Currently, this bit is always set to 1, since the DCP-Header field is only one octet long.

C/U - Compressed/Uncompressed Bit

The C/U indicates whether the data field contains compressed or uncompressed data. A value of 1 indicates compressed data (often referred to as a compressed packet), and a value of 0 indicates uncompressed data (or an uncompressed packet).

R-A - Reset-Ack

The R-A bit is used to inform the decompressing peer that the history buffer specified by the history number in the packet was in the cleared state just before the data contained in the packet was processed by the compression transformation (see section 3., Sending Compressed Datagrams). This bit MUST be set to a value of "1" to indicate a Reset-Ack, and to acknowledge a receive failure (R-R) (see section 3., Sending Compressed Datagrams). This bit is specific to the history number of the packet containing it.

R-R - Reset-Request

The R-R bit is used to request that the compressing peer clear the history buffer specified by the history number in the packet. This bit MUST be set to a value of "1" to indicate a Reset-Request, and to respond to a receive failure (R-R) (see section 3., Sending Compressed Datagrams). This bit is specific to the history number of the packet containing it.



Res - Reserved

These bits are reserved and MUST be set to 0

C/D - Control/Data

This bit is used by DCP to provide in-band negotiation in applications where out-of-band negotiation methods are not provided (i.e. Frame Relay). Since CCP provides an out of band negotiating mechanism, this feature is not used in this application. All packets MUST set this bit to a value of 0, which signifies that the packet is a data packet. (Packets containing only Reset- Requests are classified as data packets.)

#### 2.5.3. History Number

The number of the compression history which was used, ranging from 1 to the negotiated value in the History Count field.

If the negotiated History Count is less than 2, this field is removed. If the negotiated History Count is 2 or more, but less than 256, this field is 1 octet. If 256 or more histories are negotiated, this field is 2 octets, most significant octet first.

If multiple histories are used in one direction on a link, the history number field MUST be present on all packets in both directions, and sized according to the largest number of histories in either direction.

If multiple histories are used, this field MUST be present in uncompressed as well as compressed packets.

#### 2.5.4. Sequence Number

The sequence number field is one octet in length. When the check mode field is set to the "Sequence Number" or "Sequence Number + LCB" options, the sequence number field MUST be present in all data compression packets that contain a data field.

The value of the sequence number field (the sequence number of the packet) MUST begin with "1" and increment modulo 256 on successive packets that contain data fields. This number is relative to the history number used.

On receipt of a packet with the R-A bit set to "0", if the sequence number of the packet is any number other than  $(N+1) \bmod 256$ , where N is the sequence number of the last packet received

for the same history, or an initial value of "0", a receive failure for that history has occurred. The receive failure MUST be handled according to the synchronization procedure in section 3.5.

The sequence number MUST NOT be reset by the transmitter when a packet containing a Reset-Ack is sent. The decompressor MUST resynchronize its sequence number reference for the indicated history when a packet containing a Reset-Ack is received.

#### 2.5.5. Data

The data field MUST contain a single datagram in either compressed or uncompressed form, depending on the state of the C/U bit in the Header. This length of this field is always be an integer number of octets. This field is required in all packets that do not have the R-R bit set to "1".

If the C/U bit is set to "0", the data field contains the uncompressed form of the datagram.

If the C/U bit is set to "1", the form of the data field is one block of compressed data as defined in 3.2 of X3.241-1994, with the following exceptions: 1) the end marker may be followed with additional octets containing only zeros; 2) if the final octet in the block of compressed data has a value of "0", then it MAY be removed from the data field.

There is only one end marker per block of compressed data.

#### 2.5.6. Longitudinal Check Byte

The LCB field is one octet in length, and if present MUST be the last octet in the data compression packet. When the check-mode field is set to "LCB" or "Sequence Number + LCB", this field MUST be present in all packets where the data field contains compressed data. This field MUST NOT be present in data compression packets where the data field contains uncompressed data. This field contains the result of the LCB calculation, in accordance with the following paragraph.

The LCB octet is the Exclusive-OR of FF(hex) and each octet of the uncompressed datagram (prior to the compression transformation). On receipt, the receiver computes the Exclusive-OR of FF(hex) and each octet of the decompressed packet. If this value does not match the received LCB, then a receive failure for that history has occurred. The receive failure is handled according to the history synchronization procedure in section 3.5.

### 2.5.7. Compressed Data

The Stac LZS compression algorithm is Defined in ANSI X3.241-1994 [7]. The format of the compressed data is repeated here for informational purposes ONLY.

```

<Compressed Stream> := [<Compressed String>] <End Marker>
<Compressed String> := 0 <Raw Byte> | 1 <Compressed Bytes>

<Raw Byte> := <b><b><b><b><b><b><b><b>                (8-bit byte)
<Compressed Bytes> := <Offset> <Length>

<Offset> := 1 <b><b><b><b><b><b><b> |                (7-bit offset)
              0 <b><b><b><b><b><b><b><b><b><b><b><b><b><b> (11-bit offset)
<End Marker> := 110000000
<b> := 1 | 0

<Length> :=
00          = 2      1111 0110          = 14
01          = 3      1111 0111          = 15
10          = 4      1111 1000          = 16
1100        = 5      1111 1001          = 17
1101        = 6      1111 1010          = 18
1110        = 7      1111 1011          = 19
1111 0000   = 8      1111 1100          = 20
1111 0001   = 9      1111 1101          = 21
1111 0010   = 10     1111 1110          = 22
1111 0011   = 11     1111 1111 0000   = 23
1111 0100   = 12     1111 1111 0001   = 24
1111 0101   = 13     ...

```

## 3. Sending Compressed Datagrams

The reliable and efficient transport of datagrams on the data link depends on the following processes.

### 3.1. Transmitter Process

The compression operation results in either compressed or uncompressed data. When a network datagram is received, it is assigned to a particular history buffer and processed according to ANSI X3.241-1994 to form compressed data or used as is to form uncompressed data. Prior to the compression operation, if a Reset-Request is outstanding for the history buffer to be used, the buffer is cleared. In performing the compression operation, if the process mode field is set to the value None ("0"), the history MUST only be updated if the result is compressed data. If process mode field is set to the value Process-Uncompressed ("1"),

the history MUST be updated when either compressed data or uncompressed data is produced. Uncompressed data MAY be sent at any time. Uncompressed data MUST be sent if compression causes enough expansion to cause the data compression datagram size to exceed the Information field's MRU.

If the Process Mode field is set to the value None ("0") and the compressor has modified the history buffer before sending an uncompressed datagram, the history buffer MUST be cleared before the next datagram is processed.

The output of the compression operation is placed in the information field of the datagram. The C/U bit is set according to whether the data field contains compressed or uncompressed data. If the sequence number field is present according to the value of the check mode field, the sequence number counter for the applicable history number MUST be incremented and its value placed in the sequence number field. If the data field contains compressed data, and Check Mode field is set accordingly, the LCB field is present and its value is computed as specified in section 2.2.6.

Upon reception of a packet containing a Reset-Request, the transmitting compressor MUST be cleared to an initial state, which includes clearing the history buffer. If the data field of the packet containing the Reset-Request contains data, it is delivered to the local receiver as a normal data packet. In addition to the reset of the compressor, a packet MUST be transmitted with Reset-Ack bit set to 1. The data field of this packet MUST be filled with data. If no data is ready for transmission, the transmitter MUST wait until data is ready before sending the Reset-Ack.

If the history buffer is in the clear state (the history buffer contains no data bytes) prior to performing the compression operation, the resulting compressed or uncompressed packet MUST be sent with the R-A bit set to "1".

### 3.2. Receiver Process

When a data compression datagram is received from the peer, the R-R and R-A bits MUST be checked. If the R-R bit is set, the local compression engine MUST be signaled that a Reset-Request has been received for the history specified by the history number field. If the R-A bit is set, any outstanding receive failure for the specified history MUST be cleared. If no receive failure is outstanding, and the sequence number field is present, its value checked. If a receive failure has occurred, it MUST be handled according to the history resynchronization mechanism described

below, and the remainder of the datagram is discarded. If no receive failure is detected, the data is assigned to the indicated decompression history buffer and processed according to process mode field and C/U bit.

If the C/U bit is set to "1", a single octet containing the value 0x00 MUST be appended to the data field and the resulting compressed data block MUST be decompressed according to ANSI X3.241-1994. If the LCB field is present on the received datagram, an LCB for the uncompressed data MUST be computed and checked against the received LCB according to section 2.1. If a receive failure has occurred, it MUST be handled according to the History Resynchronization Mechanism described below.

If the C/U bit is set to "0" and the process mode field is set to the value Process-Uncompressed ("1"), the specified decompression history buffer MUST be updated with the received uncompressed data.

If the C/U bit is set to "0" and process mode field is set to the value None ("0"), the specified decompression history buffer MUST NOT be modified.

If the R-A bit is set to "1", the receiving decompressor MAY be reset to an initial state. (However, due to the characteristics of the Stac LZS algorithm, a decompressor reset is not required). After reset, any compressed or uncompressed data contained in the packet is processed.

On the occurrence of a receive failure, an implementation MUST transmit a packet with the R-R bit set to "1" (a Reset-Request) and with the history number matching the history that had the failure. The data field may be present if data is waiting to be transported for that history, or the R-R bit may be set in a packet transmitted without sequence number, data, or LCB fields. Once a receive failure has occurred, the data in any subsequent packets received for that history MUST be discarded until a packet containing a Reset-Ack is received. It is the responsibility of the receiver to ensure the reliability of the reset request-acknowledge mechanism. This may require the transmission of an additional Reset-Request before a Reset-Ack will be received.

### 3.3. History Maintenance

The History Count field determines the number of history buffers to be maintained for the compression protocol. For example, each history buffer could represent a separate logical connection between the data compression peers. When maintaining a history,

the peers MUST use link error detection and signaling to ensure that both the compressor and decompressor copies of each history buffer are always identical.

Setting the History Count field to the value "0" indicates that the compression is to be on a connectionless basis. In this case, a single history buffer is used and MUST be cleared at the beginning of every datagram. The compressing entity MUST set the R-A bit on all outgoing datagrams.

When the History Count field is set to the value "1", a single history buffer is maintained by each of the data compression peers. (A single logical connection.)

When the History Count field is set to a value greater than "1", separate history buffers, error detection states, and signaling states are maintained by the decompressing entity for each history. The compressing peer may transmit data on any number of separate histories, up to the value of the History Count field.

### 3.4. Anti-Expansion Mechanism

When one or more histories are negotiated and the Process Mode field is set to None ("0"), there are 2 options on how to handle packets that expand:

- 1) Send the expanded data and keep the history, thus allowing loss of current bandwidth but preserving future bandwidth on the link.
- 2) Send the uncompressed data and clear the history, thus conserving current bandwidth, but allowing possible loss of future bandwidth on the link.

When 1 or more histories are negotiated and the Process Mode field is set to Process-Uncompressed ("1"), there is an additional option:

- 3) Send the uncompressed data and do not clear the compression history; the decompressor will update its history, thus conserving the current bandwidth and future bandwidth on the link.

### 3.5. History Resynchronization Mechanism

The DCP-Header includes R-R (Reset-Request) and R-A (Reset-Ack) bits in order to provide a mechanism for indicating a receiver failure in one direction of a compressed link without affecting traffic in the other direction. A receive failure is determined

using the sequence number and/or LCB mechanism, according to the value of the check mode field.

Reset-Requests and Reset-Acks are specific to the history number of the packet containing them.

Reset-Request/Reset-Ack history synchronization signaling is provided to recover from a loss of synchronization between peers, especially in unreliable transport layers. As with all compression algorithms, the decompressor can not recover from dropped, erroneous, or mis-ordered datagrams, and will propagate errors catastrophically until both peers are reset to an initial state.

The LZS-DCP protocol provides a means to detect these error conditions: LCB for erroneous datagrams, and sequence number for dropped or mis-ordered datagrams. There is a means for correcting a loss of synchronization: clear both the failing compression and decompression histories, and follow the transmitter and receiver processes in sections 3.1. and 3.2.

#### 4. Configuration Option Format

The LZS-DCP Configuration Option negotiates the use of LZS-DCP on the link. By default or ultimate disagreement, no compression is used. This Configuration Option is used in CCP, and can be used in other negotiation mechanisms [2].

All implementations MUST support the default values.

A summary of the LZS-DCP Configuration Option format is shown below. The fields are transmitted from left to right.

0										1										2										3									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
Type										Length										History Count																			
Check Mode										Process Mode																													

Type

23

Length

6

## History Count

The History Count field is two octets, most significant octet first, and specifies the maximum number of Compression Histories.

The value 0 indicates that the implementation expects the peer to clear the Compression History at the beginning of every packet. If this value is selected, the transmitter MUST set the Reset-Ack bit of every packet that contains compressed data.

The value 1 is the default value and is used to indicate that only one history is maintained.

Other valid values range from 2 to 65535. The peer is not required to send as many histories as the implementation indicates that it can accept. However, it should be noted that resources are allocated in each peer to support the number of negotiated histories in this field.

## Check Mode

The Check Mode indicates support of LCB and/or Sequence checking. The use of check mode None (0) MUST NOT be used for history counts greater than zero.

- 0     None
- 1     LCB
- 2     Sequence Number
- 3     Sequence Number + LCB (default)

## Process Mode

The Process Mode specifies how uncompressed packets are handled. A value of None (0) indicates that uncompressed packets are not processed by the decompressor. A value of Process-Uncompressed

(1) indicates that uncompressed packets are processed by the decompressor to update the history.

- 0     None (default)
- 1     Process-Uncompressed

## Security Considerations

Security issues are not discussed in this memo.



## Acknowledgments

This document is based on, and uses much of the text of [5].

## References

- [1] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, Daydreamer, July 1994.
- [2] Rand, D., "The PPP Compression Control Protocol (CCP)", RFC 1962, June 1996.
- [3] Lempel, A., and J. Ziv, "A Universal Algorithm for Sequential Data Compression", IEEE Transactions On Information Theory, Vol. IT-23, No. 3, May 1977.
- [4] Rand, D., "PPP Reliable Transmission", RFC 1663, Novell, July 1994.
- [5] Friend, R., and W. Simpson, "PPP Stac LZS Compression Protocol", RFC 1974, August 1996.
- [6] Motorola Information Systems Group, "Data Compression Protocol (DCP) Proposal", TR-30.1 ad hoc contribution (email reflector), September 21, 1995.
- [7] ANSI X3.241-1994, "American National Standard Data Compression Method, Adaptive Coding with Sliding Window of Information Interchange".

## Chair's Address

The working group can be contacted via the current chair:

Karl Fox  
Ascend Communications  
3518 Riverside Drive, Suite 101  
Columbus, Ohio 43221

EMail: karl@ascend.com

## Authors' Addresses

Questions about this memo can also be directed to:

Kevin Schneider  
Adtran, Inc.  
901 Explorer Blvd.  
Huntsville, AL 25806

Phone: (205) 971-8024  
EMail: [kschneider@adtran.com](mailto:kschneider@adtran.com)

Robert Friend  
Stac Technology  
12636 High Bluff Drive  
San Diego, CA 92130-2093

Phone: (619) 794-4542  
EMail: [rfriend@stac.com](mailto:rfriend@stac.com)

