Network Working Group
Request for Comments: 1570
Updates: 1548
Category: Standards Track

W. Simpson, Editor
Daydreamer
January 1994

PPP LCP Extensions

Status of this Memo

Abstract

The Point-to-Point Protocol (PPP) [1] provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP defines an extensible Link Control Protocol (LCP) for establishing, configuring, and testing the data-link connection. This document defines several additional LCP features which have been suggested over the past few years.

This document is the product of the Point-to-Point Protocol Working Group of the Internet Engineering Task Force (IETF). Comments should be submitted to the ietf-ppp@ucdavis.edu mailing list.

Table of Contents

1. Additional LCP Packets

The Packet format and basic facilities are already defined for LCP [1].

Up-to-date values of the LCP Code field are specified in the most recent "Assigned Numbers" RFC [2]. This specification concerns the following values:

12	Identification
13	Time-Remaining

1.1. Identification

Description

This Code provides a method for an implementation to identify itself to its peer. This Code might be used for many diverse purposes, such as link troubleshooting, license enforcement, etc.

Identification is a Link Maintenance packet. Identification packets MAY be sent at any time, including before LCP has reached the Opened state.

The sender transmits a LCP packet with the Code field set to 12 (Identification), the Identifier field set, the local Magic-Number (if any) inserted, and the Message field filled with any desired data, but not exceeding the default MRU minus eight.

Receipt of an Identification packet causes the RXR or RUC event. There is no response to the Identification packet.

Receipt of a Code-Reject for the Identification packet SHOULD generate the RXJ+ (permitted) event.

Rationale:

This feature is defined as part of LCP, rather than as a separate PPP Protocol, in order that its benefits may be available during the earliest possible stage of the Link Establishment phase. It allows an operator to learn the identification of the peer even when negotiation is not converging. Non-LCP packets cannot be sent during the Link Establishment phase.

This feature is defined as a separate LCP Code, rather than a Configuration-Option, so that the peer need not include it with other items in configuration packet exchanges, and handle "corrected" values or "rejection", since its generation is both rare and in one direction. It is recommended that Identification packets be sent whenever a Configure-Reject is sent or received, as a final message when negotiation fails to converge, and when LCP reaches the Opened state.

A summary of the Identification packet 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Magic-Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Message ...
+-+-+-+-+-+-+-+-+
```

Code

12 for Identification

Identifier

The Identifier field MUST be changed for each Identification sent.

Length

>= 8

Magic-Number

The Magic-Number field is four octets and aids in detecting links which are in the looped-back condition. Until the Magic-Number Configuration Option has been successfully negotiated, the Magic-Number MUST be transmitted as zero. See the Magic-Number Configuration Option for further explanation.

Message

The Message field is zero or more octets, and its contents are implementation dependent. It is intended to be human readable, and MUST NOT affect operation of the protocol. It is recommended that the message contain displayable ASCII characters 32 through 126 decimal. Mechanisms for extension to other character sets are the topic of future research. The size is determined from the Length field.

Implementation Note:

The Message will usually contain such things as the sender's hardware type, PPP software revision level, and PPP product serial number, MIB information such as link speed and interface name, and any other information that the sender thinks might be useful in debugging connections. The format is likely to be different for each implementor, so that those doing serial number tracking can validate their numbers. A robust implementation SHOULD treat the Message as displayable text, and SHOULD be able to receive and display a very long Message.

1.2. Time-Remaining

Description

This Code provides a mechanism for notifying the peer of the time remaining in this session.

The nature of this information is advisory only. It is intended that only one side of the connection will send this packet (generally a "network access server"). The session is actually concluded by the Terminate-Request packet.

Time-Remaining is a Link Maintenance packet. Time-Remaining packets may only be sent in the LCP Opened state.

The sender transmits a LCP packet with the Code field set to 13 (Time-Remaining), the Identifier field set, the local Magic-Number (if any) inserted, and the Message field filled with any desired data, but not exceeding the peer's established MRU minus twelve.

Receipt of an Time-Remaining packet causes the RXR or RUC event. There is no response to the Time-Remaining packet.

Receipt of a Code-Reject for the Time-Remaining packet SHOULD generate the RXJ+ (permitted) event.

Rationale:

This notification is defined as a separate LCP Code, rather than a Configuration-Option, in order that changes and warning messages may occur dynamically during the session, and that the information might be determined after Authentication has occurred. Typically, this packet is sent when the link enters Network-Layer Protocol phase, and at regular intervals throughout the session, particularly near the end of the session.

A summary of the Time-Remaining packet 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Magic-Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Seconds-Remaining                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Message ...
+-+-+-+-+-+-+-+-+
```

Code

13 for Time-Remaining

Identifier

The Identifier field MUST be changed for each Time-Remaining sent.

Length

>= 12

Magic-Number

The Magic-Number field is four octets and aids in detecting links which are in the looped-back condition. Until the Magic-Number Configuration Option has been successfully negotiated, the Magic-Number MUST be transmitted as zero. See the Magic-Number Configuration Option for further explanation.

Seconds-Remaining

The Seconds-Remaining field is four octets and indicates the number of integral seconds remaining in this session. This 32 bit unsigned value is sent most significant octet first. A value of 0xffffffff (all ones) represents no timeout, or "forever".

Message

The Message field is zero or more octets, and its contents are implementation dependent. It is intended to be human readable, and MUST NOT affect operation of the protocol. It is recommended that the message contain displayable ASCII characters 32 through 126 decimal. Mechanisms for extension to other character sets are the topic of future research. The size is determined from the Length field.

2. Additional LCP Configuration Options

The Configuration Option format and basic options are already defined for LCP [1].

Up-to-date values of the LCP Option Type field are specified in the most recent "Assigned Numbers" RFC [2]. This document concerns the following values:

9	FCS-Alternatives
10	Self-Describing-Padding
13	Callback
15	Compound-Frames

2.1. FCS-Alternatives

Description

This Configuration Option provides a method for an implementation to specify another FCS format to be sent by the peer, or to negotiate away the FCS altogether.

This option is negotiated separately in each direction. However, it is not required that an implementation be capable of concurrently generating a different FCS on each side of the link.

The negotiated FCS values take effect only during Authentication and Network-Layer Protocol phases. Frames sent during any other phase MUST contain the default FCS.

A summary of the FCS-Alternatives Configuration Option 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |    Options    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

9

Length

3

Options

This field is one octet, and is comprised of the "logical or" of the following values:

1	Null FCS
2	CCITT 16-bit FCS
4	CCITT 32-bit FCS

Implementation Note:

For most PPP HDLC framed links, the default FCS is the CCITT 16-bit FCS. Some framing techniques and high speed links may use another format as the default FCS.

2.1.1. LCP considerations

The link can be subject to loss of state, and the LCP can re-negotiate at any time. When the LCP begins renegotiation or termination, it is recommended that the LCP Configure-Request or Terminate-Request packet be sent with the last negotiated FCS, then change to the default FCS, and a duplicate LCP packet is sent with the default FCS. The Identifier field SHOULD NOT be incremented for each such duplicate packet.

On receipt of a LCP Configure-Request or Terminate-Request packet, the implementation MUST change to the default FCS for both transmission and reception. If a Request packet is received which contains a duplicate Identifier field, a new reply MUST be generated.

Implementation Notes:

The need to send two packets is only necessary after the Alternative-FCS has already been negotiated. It need not occur during state transitions when there is a natural indication that the default FCS is in effect, such as the Down and Up events. It is necessary to send two packets in the Ack-Sent and Opened states, since the peer could mistakenly believe that the link has Opened.

It is possible to send a single 48-bit FCS which is a combination of the 16-bit and 32-bit FCS. This may be sent instead of sending the two packets described above. We have not standardized this procedure because of intellectual property concerns. If such a 48-bit FCS is used, it MUST only be used for LCP packets.

2.1.2. Null FCS

The Null FCS SHOULD only be used for those network-layer and transport protocols which have an end-to-end checksum available, such as TCP/IP, or UDP/IP with the checksum enabled. That is, the Null FCS option SHOULD be negotiated together with another non-null FCS option in a heterogeneous environment.

When a configuration (LCP or NCP) or authentication packet is sent, the FCS MUST be included. When a configuration (LCP or NCP) or authentication packet is received, the FCS MUST be verified.

There are several cases to be considered:

Null FCS alone

The sender generates the FCS for those frames which require the FCS before sending the frame.

When a frame is received, it is not necessary to check the FCS before demultiplexing. Any FCS is treated as padding.

Receipt of an Authentication or Control packet would be discovered after passing the frame to the demultiplexer. Verification of the FCS can easily be accomplished using one of the software algorithms defined in "PPP in HDLC Framing" [3] (16-bit FCS) and Appendix A (32-bit FCS).

Null FCS with another FCS, using software

This is similar to the above case.

Those packets which are required to have the FCS (Authentication, Control, or Network-Protocols lacking a checksum) are checked using software after demultiplexing. Packets which fail the FCS test are discarded as usual.

Null FCS with another FCS, using hardware

A flag is passed with the frame, indicating whether or not it has passed the hardware FCS check. The incorrect FCS MUST be passed with the rest of the data. The frame MUST NOT be discarded until after demultiplexing, and only those frames that require the FCS are discarded.

All three FCS forms (Null, 16 and 32) may be used concurrently on different frames when using software. That is probably not possible with most current hardware.

2.2. Self-Describing-Padding

Description

This Configuration Option provides a method for an implementation to indicate to the peer that it understands self-describing pads when padding is added at the end of the PPP Information field.

This option is most likely to be used when some protocols, such as network-layer or compression protocols, are configured which require detection and removal of any trailing padding. Such special protocols are identified in their respective documents.

If the option is Rejected, the peer MUST NOT add any padding to the identified special protocols, but MAY add padding to other protocols.

If the option is Ack'd, the peer MUST follow the procedures for adding self-describing pads, but only to the specifically identified protocols. The peer is not required to add any padding to other protocols.

Implementation Notes:

This is defined so that the Reject handles either case where the peer does not generate self-describing pads. When the peer never generates padding, it may safely Reject the option. When the peer does not understand the option, it also will not successfully configure a special protocol which requires elimination of pads.

While some senders might only be capable of adding padding to every protocol or not adding padding to any protocol, by design the receiver need not examine those protocols which do not need the padding stripped.

To avoid unnecessary configuration handshakes, an implementation which generates padding, and has a protocol configured which requires the padding to be known, SHOULD include this Option in its Configure-Request, and SHOULD

Configure-Nak with this Option when it is not present in the peer’s Request.

Each octet of self-describing pad contains the index of that octet. The first pad octet MUST contain the value one (1), which indicates the Padding Protocol to the Compound-Frames option. After removing the FCS, the final pad octet indicates the number of pad octets to remove. For example, three pad octets would contain the values 1, 2, 3.

The Maximum-Pad-Value (MPV) is also negotiated. Only the values 1 through MPV are used. When no padding would otherwise be required, but the final octet of the PPP Information field contains the value 1 through MPV, at least one self-describing pad octet MUST be added to the frame. If the final octet is greater than MPV, no additional padding is required.

Implementation Notes:

If any of the pad octets contain an incorrect index value, the entire frame SHOULD be silently discarded. This is intended to prevent confusion with the FCS-Alternatives option, but might not be necessary in robust implementations.

Since this option is intended to support compression protocols, the Maximum-Pad-Value is specified to limit the likelihood that a frame may actually become longer.

A summary of the Self-Describing-Padding Configuration Option 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |    Maximum    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

10

Length

3

Maximum

This field specifies the largest number of padding octets which may be added to the frame. The value may range from 1 to 255, but values of 2, 4, or 8 are most likely.

2.3. Callback

Description

This Configuration Option provides a method for an implementation to request a dial-up peer to call back. This option might be used for many diverse purposes, such as savings on toll charges.

When Callback is successfully negotiated, and authentication is complete, the Authentication phase proceeds directly to the Termination phase, and the link is disconnected.

Then, the peer re-establishes the link, without negotiating Callback.

Implementation Notes:

A peer which agrees to this option SHOULD request the Authentication-Protocol Configuration Option. The user information learned during authentication can be used to determine the user location, or to limit a user to certain locations, or merely to determine whom to bill for the service.

Authentication SHOULD be requested in turn by the implementation when it is called back, if mutual authentication is desired.

A summary of the Callback 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     |   Operation   |  Message ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

13

Length

>= 3

Operation

The Operation field is one octet and indicates the contents of the Message field.

0	location is determined by user authentication
1	Dialing string, the format and contents of which assumes configuration knowledge of the specific device which is making the callback.
2	Location identifier, which may or may not be human readable, to be used together with the authentication information for a database lookup to determine the callback location.
3	E.164 number.
4	Distinguished name.

Message

The Message field is zero or more octets, and its general contents are determined by the Operation field. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets. The size is determined from the Length field.

It is intended that only an authorized user will have correct site specific information to make use of the Callback. The codification of the range of allowed usage of this field is outside the scope of this specification.

2.4. Compound-Frames

Description

This Configuration Option provides a method for an implementation to send multiple PPP encapsulated packets within the same frame. This option might be used for many diverse purposes, such as savings on toll charges.

Only those PPP Protocols which have determinate lengths or integral length fields may be aggregated into a compound frame.

When Compound-Frames is successfully negotiated, the sender MAY add additional packets to the same frame. Each packet is immediately followed by another Protocol field, with its attendant datagram.

When padding is added to the end of the Information field, the procedure described in Self-Describing-Padding is used. Therefore, this option MUST be negotiated together with the Self-Describing-Padding option.

If the FCS-Alternatives option has been negotiated, self describing padding MUST always be added. That is, the final packet MUST be followed by a series of octets, the first of which contains the value one (1).

On receipt, the first Protocol field is examined, and the packet is processed as usual. For those datagrams which have a determinate length, the remainder of the frame is returned to the demultiplexor. Each succeeding Protocol field is processed as a separate packet. This processing is complete when a packet is processed which does not have a determinate length, when the remainder of the frame is empty, or when the Protocol field is determined to have a value of one (1).

The PPP Protocol value of one (1) is reserved as the Padding Protocol. Any following octets are removed as padding.

A summary of the Compound-Frames Option format is shown below. The fields are transmitted from left to right.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type

   15

Length

   2

2.4.1. LCP considerations

During initial negotiation, the Compound-Frames option can be used to minimize the negotiation latency, by reducing the number of frames exchanged.

The first LCP Configure-Request packet is sent as usual in a single frame, including the Self-Describing-Padding and Compound-Frames options.

The peer SHOULD respond with a Configure-Ack, followed in a compound frame by its LCP Configure-Request, and any NCP Configure-Requests desired.

Upon receipt, the local implementation SHOULD process the Configure-Ack as usual. Since the peer has agreed to send compound frames, the implementation MUST examine the remainder of the frame for additional packets. If the peer also specified the Self-Describing-Padding and Compound-Frames options in its Configure-Request, the local implementation SHOULD retain its Configure-Ack, and further NCP configuration packets SHOULD be added to the return frame.

Together with the peer’s final return frame, the minimum number of frames to complete configuration is 4.

A. Fast Frame Check Sequence (FCS) Implementation

A.1. 32-bit FCS Computation Method

The following code provides a table lookup computation for calculating the 32-bit Frame Check Sequence as data arrives at the interface.

```
   /*
    * u32 represents an unsigned 32-bit number.  Adjust the typedef for
    * your hardware.
    */
   typedef unsigned long u32;

   static u32 fcstab_32[256] =
      {
      0x00000000, 0x77073096, 0xee0e612c, 0x990951ba,
      0x076dc419, 0x706af48f, 0xe963a535, 0x9e6495a3,
      0x0edb8832, 0x79dcb8a4, 0xe0d5e91e, 0x97d2d988,
      0x09b64c2b, 0x7eb17cbd, 0xe7b82d07, 0x90bf1d91,
      0x1db71064, 0x6ab020f2, 0xf3b97148, 0x84be41de,
      0x1adad47d, 0x6ddde4eb, 0xf4d4b551, 0x83d385c7,
      0x136c9856, 0x646ba8c0, 0xfd62f97a, 0x8a65c9ec,
      0x14015c4f, 0x63066cd9, 0xfa0f3d63, 0x8d080df5,
      0x3b6e20c8, 0x4c69105e, 0xd56041e4, 0xa2677172,
      0x3c03e4d1, 0x4b04d447, 0xd20d85fd, 0xa50ab56b,
      0x35b5a8fa, 0x42b2986c, 0xdbbbc9d6, 0xacbcf940,
      0x32d86ce3, 0x45df5c75, 0xdcd60dcf, 0xabd13d59,
      0x26d930ac, 0x51de003a, 0xc8d75180, 0xbfd06116,
      0x21b4f4b5, 0x56b3c423, 0xcfba9599, 0xb8bda50f,
      0x2802b89e, 0x5f058808, 0xc60cd9b2, 0xb10be924,
      0x2f6f7c87, 0x58684c11, 0xc1611dab, 0xb6662d3d,
      0x76dc4190, 0x01db7106, 0x98d220bc, 0xefd5102a,
      0x71b18589, 0x06b6b51f, 0x9fbfe4a5, 0xe8b8d433,
      0x7807c9a2, 0x0f00f934, 0x9609a88e, 0xe10e9818,
      0x7f6a0dbb, 0x086d3d2d, 0x91646c97, 0xe6635c01,
      0x6b6b51f4, 0x1c6c6162, 0x856530d8, 0xf262004e,
      0x6c0695ed, 0x1b01a57b, 0x8208f4c1, 0xf50fc457,
      0x65b0d9c6, 0x12b7e950, 0x8bbeb8ea, 0xfcb9887c,
      0x62dd1ddf, 0x15da2d49, 0x8cd37cf3, 0xfbd44c65,
      0x4db26158, 0x3ab551ce, 0xa3bc0074, 0xd4bb30e2,
      0x4adfa541, 0x3dd895d7, 0xa4d1c46d, 0xd3d6f4fb,
      0x4369e96a, 0x346ed9fc, 0xad678846, 0xda60b8d0,
      0x44042d73, 0x33031de5, 0xaa0a4c5f, 0xdd0d7cc9,
      0x5005713c, 0x270241aa, 0xbe0b1010, 0xc90c2086,
      0x5768b525, 0x206f85b3, 0xb966d409, 0xce61e49f,
      0x5edef90e, 0x29d9c998, 0xb0d09822, 0xc7d7a8b4,
      0x59b33d17, 0x2eb40d81, 0xb7bd5c3b, 0xc0ba6cad,
      0xedb88320, 0x9abfb3b6, 0x03b6e20c, 0x74b1d29a,
      0xead54739, 0x9dd277af, 0x04db2615, 0x73dc1683,
      0xe3630b12, 0x94643b84, 0x0d6d6a3e, 0x7a6a5aa8,
      0xe40ecf0b, 0x9309ff9d, 0x0a00ae27, 0x7d079eb1,
      0xf00f9344, 0x8708a3d2, 0x1e01f268, 0x6906c2fe,
      0xf762575d, 0x806567cb, 0x196c3671, 0x6e6b06e7,
      0xfed41b76, 0x89d32be0, 0x10da7a5a, 0x67dd4acc,
      0xf9b9df6f, 0x8ebeeff9, 0x17b7be43, 0x60b08ed5,
      0xd6d6a3e8, 0xa1d1937e, 0x38d8c2c4, 0x4fdff252,
      0xd1bb67f1, 0xa6bc5767, 0x3fb506dd, 0x48b2364b,
      0xd80d2bda, 0xaf0a1b4c, 0x36034af6, 0x41047a60,
      0xdf60efc3, 0xa867df55, 0x316e8eef, 0x4669be79,
      0xcb61b38c, 0xbc66831a, 0x256fd2a0, 0x5268e236,
      0xcc0c7795, 0xbb0b4703, 0x220216b9, 0x5505262f,
      0xc5ba3bbe, 0xb2bd0b28, 0x2bb45a92, 0x5cb36a04,
      0xc2d7ffa7, 0xb5d0cf31, 0x2cd99e8b, 0x5bdeae1d,
      0x9b64c2b0, 0xec63f226, 0x756aa39c, 0x026d930a,
      0x9c0906a9, 0xeb0e363f, 0x72076785, 0x05005713,
      0x95bf4a82, 0xe2b87a14, 0x7bb12bae, 0x0cb61b38,
      0x92d28e9b, 0xe5d5be0d, 0x7cdcefb7, 0x0bdbdf21,
      0x86d3d2d4, 0xf1d4e242, 0x68ddb3f8, 0x1fda836e,
      0x81be16cd, 0xf6b9265b, 0x6fb077e1, 0x18b74777,
      0x88085ae6, 0xff0f6a70, 0x66063bca, 0x11010b5c,
      0x8f659eff, 0xf862ae69, 0x616bffd3, 0x166ccf45,
      0xa00ae278, 0xd70dd2ee, 0x4e048354, 0x3903b3c2,
      0xa7672661, 0xd06016f7, 0x4969474d, 0x3e6e77db,
      0xaed16a4a, 0xd9d65adc, 0x40df0b66, 0x37d83bf0,
      0xa9bcae53, 0xdebb9ec5, 0x47b2cf7f, 0x30b5ffe9,
      0xbdbdf21c, 0xcabac28a, 0x53b39330, 0x24b4a3a6,
      0xbad03605, 0xcdd70693, 0x54de5729, 0x23d967bf,
      0xb3667a2e, 0xc4614ab8, 0x5d681b02, 0x2a6f2b94,
      0xb40bbe37, 0xc30c8ea1, 0x5a05df1b, 0x2d02ef8d
      };

   #define PPPINITFCS32  0xffffffff   /* Initial FCS value */
   #define PPPGOODFCS32  0xdebb20e3   /* Good final FCS value */

   /*
    * Calculate a new FCS given the current FCS and the new data.
    */
   u32 pppfcs32(fcs, cp, len)
       register u32 fcs;
       register unsigned char *cp;
       register int len;
       {
       ASSERT(sizeof (u32) == 4);
       ASSERT(((u32) -1) > 0);
       while (len--)
```

```
                fcs = (((fcs) >> 8) ^ fcstab_32[((fcs) ^ (*cp++)) & 0xff]);

            return (fcs);
            }

       /*
        * How to use the fcs
        */
       tryfcs32(cp, len)
           register unsigned char *cp;
           register int len;
       {
           u32 trialfcs;

           /* add on output */
           trialfcs = pppfcs32( PPPINITFCS32, cp, len );
           trialfcs ^= 0xffffffff;             /* complement */
           cp[len] = (trialfcs & 0x00ff);      /* Least significant byte first *
/
           cp[len+1] = ((trialfcs >>= 8) & 0x00ff);
           cp[len+2] = ((trialfcs >>= 8) & 0x00ff);
           cp[len+3] = ((trialfcs >> 8) & 0x00ff);

           /* check on input */
           trialfcs = pppfcs32( PPPINITFCS32, cp, len + 4 );
           if ( trialfcs == PPPGOODFCS32 )
               printf("Good FCS\n");
       }
```

Security Considerations

Security issues are briefly discussed in sections concerning the Callback Configuration Option.

References

[1]   Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", RFC 1548, Daydreamer, December 1993.

[2]   Reynolds, J., and J. Postel, "Assigned Numbers", STD 2, RFC 1340, USC/Information Sciences Institute, July 1992.

[3]   Simpson, W., Editor, "PPP in HDLC Framing", RFC 1549, Daydreamer, December 1993.

Acknowledgments

The Identification feature was suggested by Bob Sutterfield (Morning Star Technologies).

The Time-Remaining feature was suggested by Brad Parker (FCR).

Some of the original text for FCS-Alternatives was provided by Arthur Harvey (then of DEC). The Null FCS was requested by Peter Honeyman (UMich). The 32-bit FCS example code was provided by Karl Fox (Morning Star Technologies).

Self-Describing-Padding was suggested and named by Fred Baker (ACC).

Compound-Frames was suggested by Keith Sklower (Berkeley).

Special thanks to Morning Star Technologies for providing computing resources and network access support for writing this specification.

Chair’s Address

The working group can be contacted via the current chair:

Fred Baker
Advanced Computer Communications
315 Bollay Drive
Santa Barbara, California 93117

EMail: fbaker@acc.com

Editor’s Address

Questions about this memo can also be directed to:

William Allen Simpson
Daydreamer
Computer Systems Consulting Services
1384 Fontaine
Madison Heights, Michigan 48071

EMail: Bill.Simpson@um.cc.umich.edu
bsimpson@MorningStar.com