Technical Document – Confidential

TEST SPECIFICATION

PPP

Document Number:	8441.408.99.010
Version:	0.12
Status:	Draft
Approval Authority:	
Creation Date:	2000-Jun-14
Last changed:	2015-Mar-08 by XGUTTEFE
File Name:	ppp.doc

Important Notice

Change History

Date	Changed by	Approved by	Version	Status	Notes
2000-Jun-14	Stefan Grundmann		0.1		1
2000-Aug-09	STW		0.2		2
2000-Oct-18	STW		0.3		3
2001-Jan-22	STW		0.4		4
2001-Jun-22	XOF		0.5		5
2001-Jul-03	XOF		0.6		6
2001-Jul-12	STW		0.7		7
2001-Sep-20	XOF		0.8		8

2002-Jan-25	STW		0.9		9
2002-Apr-25	TVO		0.10		10
2002-Oct-25	STW		0.11		11
2003-May-26	XGUTTEFE		0.12	Draft	

Notes:

1. Initial version
2. Document number changed
3. Server mode test cases added
4. Special cases of IP address negotiation added
5. Channel ID and connect primitive added
6. Test case PPP044 and PPP011 corrected
7. Test of PCO processing added
8. Build DTILIB functionality
9. different Flow Control behavior because PPP stores one IP packet
10. New tests for closed prot channel (015, 020B, 030B, 040B, 055, 051B, 060B) and changed order of primitives in transparent acknowledgement (014)
11. Cause concept

Table of Contents

1.1 References 7
1.2 Abbreviations 9
1.3 Terms 12

2 Overview 12

2.1 GRR (RLC/MAC) – Radio Link Control/Medium Access Control 13
2.2 LLC – Logical Link Control 13
2.3 GMM – GPRS Mobility Management 13
2.4 SM – Session Management 13
2.5 SNDCP - Subnetwork Dependant Convergence Protocol 13
2.6 GACI – GPRS Application Control Interface 13
2.7 USART - Universal Synchronous Asynchronous Receiver Transmitter Driver 13
2.8 TOM – Tunnelling of Messages 13

3 Parameters 14

3.1 Primitive elements 14
3.2 Struct elements 14
3.3 Fields 14
3.3.1 LCP Configure Request 14
3.3.2 LCP Configure Ack 29
3.3.3 LCP Configure Reject 38
3.3.4 LCP Configure NAK 42
3.3.5 LCP Terminate Request 45
3.3.6 LCP Terminate Ack 46
3.3.7 LCP Code Reject 47
3.3.8 LCP Protocol Reject 49
3.3.9 LCP Identification 57
3.3.10 CHAP Challenge 60
3.3.11 CHAP Response 61
3.3.12 CHAP Success 63
3.3.13 PAP Authentication Request 64
3.3.14 PAP Authentication NAK 66
3.3.15 PAP Authentication Ack 67
3.3.16 CCP Configure Request 69
3.3.17 IPCP Configure Request 71
3.3.18 IPCP Configure Reject 85
3.3.19 IPCP Configure NAK 89
3.3.20 IPCP Configure Ack 93
3.3.21 IP packets 100
3.3.22 IP frames 103
3.3.23 HDLC fragments 111
3.3.24 Invalid frames 114
3.3.25 Empty sdu 116
3.3.26 PCO lists 116
3.4 Declarations 123
3.5 Arrays 123
3.6 Structs 124

4 TEST CASES 126

4.1 Setup (PPP001 - PPP010) 126

4.1.1 PPP001: Setup 126
4.2 Link Establishment Requests (PPP011 – PPP019) 126
4.2.1 PPP011: Link establishment in client mode (default parameters) 126
4.2.2 PPP012: Link establishment in client mode (PAP) 128
4.2.3 PPP013: Link establishment in server mode 130
4.2.4 PPP014: Link establishment in transparent mode 132
4.2.5 PPP015: Link establishment in client mode (PAP), only peer dti channel opened 134
4.3 LCP Establishment (PPP020-PPP029) 136
4.3.1 PPP020: LCP Establishment (Client, PAP) 136
4.3.2 PPP021: LCP Establishment (CLient, PAP, ACFC, PFC) 137
4.3.3 PPP022: LCP Establishment (CLient, PAP, ACFC, PFC) with magic number request 138
4.3.4 PPP023: LCP Establishment (Server, CHAP, ACFC, PFC) with magic number request 140
4.3.5 PPP024: LCP Establishment (Server) re-request of rejected configuration options ... 142
4.4 Authentication Procedure (PPP030-PPP039) 145
4.4.1 PPP030: PAP Authentication in Client Mode 145
4.4.2 PPP031: PAP Authentication in Client Mode (ACFC, PFC) 146
4.4.3 PPP032: PAP authtentication failed (client mode) 147
4.4.4 PPP033: CHAP Authentication in Server Mode (ACFC, PFC) with LCP Identification 149
4.4.5 PPP034: PAP Authentication in Server Mode (ACFC, PFC) with LCP Identification .. 151
4.5 IPCP Establishment (PPP40-PPP049) 153
4.5.1 PPP040: Normal IPCP Address negotiation 153
4.5.2 PPP041: Normal IPCP Address negotiation (ACFC, PFC) 155
4.5.3 PPP042: CCP request from server during ICPC procedure 157
4.5.4 PPP043: IPCP configuration in server mode with CCP 160
4.5.5 PPP044: PDP activation will be rejected by the network 165
4.6 Link Establishment Confirmation (PPP050-PPP059) 167
4.6.1 PPP050: Confiramtion of link establishment in client mode 167
4.6.2 PPP051: Open Flow Control for Downlink 168
4.6.3 PPP052: Confiramtion of link establishment in client mode (ACFC, PFC) 169
4.6.4 PPP053: Open Flow Control for Downlink (ACFC, PFC) 169
4.6.5 PPP055: Confirmation of link establishment in client mode and acknowledgement of prot dti connection 170
4.7 Uplink Transfer (PPP060-PPP69) 171
4.7.1 PPP060: IP Packet in HDLC Frame 171
4.7.2 PPP061: IP Packet in two HDLC Frames 173
4.7.3 PPP062: Two Packets in three Frames 174
4.7.4 PPP063: IP Packet in HDLC Frame (ACFC, PFC) 176
4.7.5 PPP064: IP Packet in two HDLC Frames (ACFC, PFC) 177
4.7.6 PPP065: Two Packets in three Frames (ACFC, PFC) 178
4.7.7 PPP066: Three Packets in one Frames 180
4.7.8 PPP067: Three Packets in one Frame (ACFC,PFC) 181
4.7.9 PPP068: TCP Packets without Header Compression Negotiation (ACFC,PFC) 183
4.7.10 PPP069: TCP Packets with Header Compression Negotiation (ACFC,PFC) 184
4.8 Downlink Transfer (PPP070-PPP79) 186
4.8.1 PPP070: Simple Downlink transfer 186
4.8.2 PPP071: Simple Downlink transfer (ACFC, PFC) 187
4.9 PPP Termination (PPP080-PPP083) 188
4.9.1 PPP080: Termination initiated by MMI, lower layer available 188
4.9.2 PPP081: Termination initiated by MMI, lower layer not available 189
4.9.3 PPP082: Termination initiated by PPP Peer 190
4.10 Uplink Transfer with invalid packets (PPP090-PPP099) 191
4.10.1 PPP090: one frame with invalid control field received 191
4.10.2 PPP091: one frame with invalid protocol field received 193

4.10.3 PPP092: one frame with invalid address field received 194
4.10.4 PPP093: one frame with invalid FSC received 195
4.11 Modification of PPP link (PPP100-PPP109) 197
4.11.1 PPP100: turn header compression off 197
4.12 Transparent packet transfer (PPP110-PPP119) 200
4.12.1 PPP110: Transparent packet transfer uplink and downlink 200
4.13 Link termination in transparent mode 203
4.13.1 PPP120: Termination of transparent link 203
4.14 Link reestablishment 204
4.14.1 PPP130: Reestablishment in client mode 204
4.15 IP address negotiation (special cases) 207
4.15.1 PPP140: Server IP address in client mode allowed 207
4.15.2 PPP141: Rejection of dynamic server IP address in client mode 209
4.15.3 PPP142: Allow always static DNS addresses in server mode 212
4.15.4 PPP143: Reject dynamic DNS addresses in server mode if network does not answer 215

Appendices 219

A. Acronyms 219
B. Glossary 219

List of Figures and Tables

List of References

[ISO 9000:2000] International Organization for Standardization. Quality management systems - Fundamentals and vocabulary. December 2000

1.1 References

[1] GSM 05.02 version 8.0.0 Release 1999
Digital cellular telecommunications system (Phase 2+);
Multiplexing and multiple access on the radio path

[2] GSM 04.60 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Mobile Station (MS) - Base Station System (BSS) interface;
Radio Link Control/ Medium Access Control (RLC/MAC) protocol

[3] GSM 04.08 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Mobile radio interface layer 3 specification

[4] GSM 03.64 version 6.1.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Overall description of the GPRS radio interface; Stage 2

[5] GSM 03.60 version 6.3.1 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Service description; Stage 2

[6] GSM 04.07 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Mobile radio interface signalling layer 3; General aspects

[7] GSM 04.64 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
General Packet Radio Service (GPRS);
Mobile Station - Serving GPRS Support Node (MS-SGSN)
Logical Link Control (LLC) layer specification

[8] GSM 05.08 version 6.4.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Radio subsystem link control

[9] GSM 05.10 version 6.3.0 Release 1997
Digital cellular telecommunications system (Phase 2+);
Radio subsystem synchronization

[10] GSM 03.20 TS 100 929: July 1998 (GSM 03.20 version 6.0.1)
Security related network functions, ETSI

[11] Draft GSM 03.22: August 1998 (GSM 03.22 version 6.1.0)
Functions related to Mobile Station (MS) in idle mode and group receive mode, ETSI

[12] GSM 04.65 V6.3.0: Subnetwork Dependant Convergence Protocol
ETSI, March 1999

[13] ITU-T V42bis ITU-T, Recommendation V.42 bis 1990

[14] GSM 09.60 GPRS Tunneling Protocol (GTP) across the Gn and Gp Interface

[15] RFC 1661 IETF STD 51 July 1994
The Point-to-Point Protocol (PPP)

[16] RFC 1662 IETF STD 51 July 1994
PPP in HDLC-like Framing

[17] RFC 1570 January 1994
PPP LCP Extensions

[18] RFC 1989 August 1996
PPP Link Quality Monitoring

[19] RFC 1332 May 1992
The PPP Internet Protocol Control Protocol (IPCP)

[20] RFC 1877 December 1995
PPP IPCP Extensions for Name Server Addresses

[21] RFC 2153 May 1997
PPP Vendor Extensions

[22] RFC 1334 October 1992
PPP Authentication Protocols (for Password Authentication Protocol only)

[23] RFC 1994 August 1996
PPP Challenge Handshake Authentication Protocol (CHAP)

[24] TIA/EIA-136-370
Packet-Data Services – Enhanced General Packet Radio for TIA/EIA-136 (EGPRS-136) - Overview, Telecommunications Industry Association

[25] TIA/EIA-136-376
Packet-Data Services – EGPRS-136 Mobility Management, Telecommunications Industry Association

[26] TIA/EIA-136-972
Packet-Data Services – Stage 2 Description, Telecommunications Industry Association

1.2 Abbreviations

ACI	Application Control Interface
AGCH	Access Grant Channel
AT	Attention sequence "AT" to indicate valid commands of the ACI
BCCH	Broadcast Control Channel
BS	Base Station
BSIC	Base Station Identification Code
C/R	Command/Response
C1	Path Loss Criterion
C2	Reselection Criterion
CBCH	Cell Broadcast Channel
CBQ	Cell Bar Qualify
CC	Call Control
CCCH	Common Control Channel
CCD	Condat Coder Decoder
CCI	Compression and Ciphering Interface
CHAP	Challenge Handshake Authentication Protocol
CKSN	Ciphering Key Sequence Number
CRC	Cyclic Redundancy Check
DCCH	Dedicated Control Channel
DCOMP	Identifier of the user data compression algorithm used for the N-DPU
DISC	Disconnect Frame
DL	Data Link Layer
DM	Disconnected Mode Frame
DTX	Discontinuous Transmission
E	Extension bit
EA	Extension Bit Address Field
EL	Extension Bit Length Field
EMMI	Electrical Man Machine Interface
F	Final Bit
FACCH	Fast Associated Control Channel
FHO	Forced Handover
GACI	GPRS Application Control Interface
GMM	GPRS Mobility Management
GP	Guard Period
GRR	GPRS RR
GSM	Global System for Mobile Communication
HDLC	High-level Data Link Control
HISR	High level Interrupt Service Routine
HPLMN	Home Public Land Mobile Network
I	Information Frame
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
ITU	International Telecommunication Union
IWF	Interworking Function
Kc	Ciphering Key
L	Length Indicator
LAI	Location Area Information
LCP	Link Control Protocol
LISR	Low level Interrupt Service Routine

LLC	Logical Link Control
LPD	Link Protocol Discriminator
LQM	Link Quality Monitoring
M	More bit used to indicate the last segment of N-DPU
MAC	Medium Access Control
MCC	Mobile Country Code
MM	Mobility Management
MMI	Man Machine Interface
MNC	Mobile Network Code
MS	Mobile Station
MT	Mobile Termination
N(R)	Receive Number
N(S)	Send Number
NC	Network Control
NCC	National Colour Code
NCP	Network Control Protocol
NECI	New Establishment Causes included
N-PDU	Network Protocol Data Unit
NSAPI	Network Layer Service Access Point Identifier
OTD	Observed Time Difference
P	Poll Bit
P/F	Poll/Final Bit
PACCH	Packet Associated Control Channel
PAP	Password Authentication Protocol
PBCCH	Packet BCCH
PCCCH	Packet CCCH
PCOMP	Identifier of the protocol control information compression algorithm used for the N-DPU
PDCH	Packet Data Channel
PDP	Packet Data Protocol e.g. IP or X.25
PDTCH	Packet Data Traffic Channel
PRACH	Packet RACH
PSI	Packet System Information
PCH	Paging Channel
PCO	Point of Control and Observation
PDU	Protocol Data Unit
PL	Physical Layer
PLMN	Public Land Mobile Network
PPC	Packet Physical Convergence
PPP	Point-to-Point Protocol
PTP	Point to Point
QoS	Quality of Service
RACH	Random Access Channel
REJ	Reject Frame
RLC	Radio Link Control
RNR	Receive Not Ready Frame
RR	Radio Resource Management
RR	Receive Ready Frame
RTD	Real Time Difference
RTOS	Real Time Operating System
SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated Control Channel
SAP	Service Access Point
SAPI	Service Access Point Identifier
SDCCH	Stand alone Dedicated Control Channel
SDU	Service Data Unit
SGSN	Serving GPRS Support Node

SIM	Subscriber Identity Module
SM	Session Management
SMS	Short Message Service
SMSCB	Short Message Service Cell Broadcast
SNDCP	Subnetwork Dependant Convergence Protocol
SNSM	SNDCP-SM
SS	Supplementary Services
TAP	Test Application Program
TBF	Temporary Block Flow
TCH	Traffic Channel
TCH/F	Traffic Channel Full Rate
TCH/H	Traffic Channel Half Rate
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TE	Terminal Equipment - e. g. a PC
TFI	Temporary Flow Identifier
TLLI	Temporary Logical Link Identifier
TMSI	Temporary Mobile Subscriber Identity
TOM	Tunnelling of Messages
TQI	Temporary Queuing Identifier
UA	Unnumbered Acknowledgement Frame
UART	Universal Asynchronous Receiver Transmitter
UI	Unnumbered Information Frame
USF	Uplink State Flag
V(A)	Acknowledgement State Variable
V(R)	Receive State Variable
V(S)	Send State Variable
VPLMN	Visited Public Land Mobile Network

1.3 Terms

Entity:	Program which executes the functions of a layer
Message:	A message is a data unit which is transferred between the entities of the same layer (peer-to-peer) of the mobile and infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements.
Primitive:	A primitive is a data unit which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code which identifies the primitive and its parameters.
Service Access Point:	A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure).

2 Overview

The Protocol Stacks are used to define the functionality of the GSM protocols for interfaces. The GSM specifications are normative when used to describe the functionality of interfaces, but the stacks and the subdivision of protocol layers does not imply or restrict any implementation.

The protocol stack for GPRS consists of several entities. Each entity has one ore more service access points, over which the entity provides a service for the upper entity.

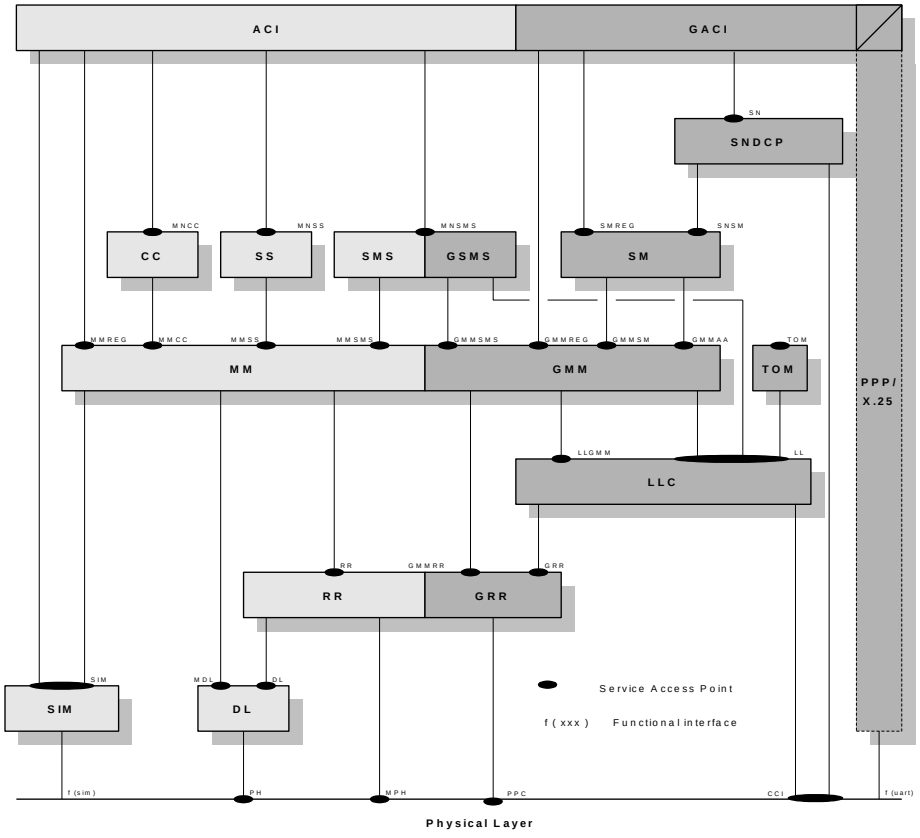


Figure 2-1: Architecture of the GSM/GPRS protocol stack

The information units passed via the SAPs are called primitives and consists of an operation code and several parameters. See the Users Guide for details.

The entities of the GPRS protocol stack are:

2.1 GRR (RLC/MAC) – Radio Link Control/Medium Access Control

This layer contains two functions: The Radio Link Control function provides a radio-solution-dependent reliable link. The Medium Access Control function controls the access signalling (request and grant) procedures for the radio channel, and the mapping of LLC frames onto the GSM physical channel.

2.2 LLC – Logical Link Control

The LLC entity provides multiple highly reliable logical links for asynchronous data transfer between the MS and the network. It supports variable-length information frames, acknowledged and unacknowledged data transfer, flow and sequence control, error detection and recovery, notification of unrecoverable errors, user identity confidentiality, and ciphering of user and signaling data.

2.3 GMM – GPRS Mobility Management

The GMM entity provides procedures for the mobility of the MS, such as informing the network of its present location, and user identity confidentiality. It manages the GMM context (attach, detach, routing area updating), supports security functions such as authentication of user and MS, controls ciphering of data, and initiates the response to paging messages.

2.4 SM – Session Management

The main function of the session management (SM) is to support PDP context handling of the user terminal. Session Management activates, modifies and deletes the contexts for packet data protocols (PDP). Session Management services are provided at the SMREG-SAP and the SNSM-SAP for anonymous and non-anonymous access. The non-anonymous and anonymous access procedures for PDP context activation and PDP context deactivation are available at the SMREG-SAP. In addition there exists a PDP context modification for non-anonymous PDP contexts.

2.5 SNDCP - Subnetwork Dependant Convergence Protocol

SNDCP carries out all functions related to transfer of Network layer Protocol Data Units (N-PDUs) over GPRS in a transparent way. SNDCP helps to improve channel efficiency by means of compression techniques. The set of protocol entities above SNDCP consists of commonly used network protocols. They all use the same SNDCP entity, which then performs multiplexing of data coming from different sources to be sent using the service provided by the LLC layer.

2.6 GACI – GPRS Application Control Interface

The GACI is the GPRS extension of the ACI. It is specified in GSM 07.07 and 07.60. It is responsible for processing of the GPRS related AT Commands to setup, activate and deactivate the PDP context parameter. It also provides functionality for the interworking between GMM/SM/SNDCP and a packet oriented protocol like PPP.

2.7 USART - Universal Synchronous Asynchronous Receiver Transmitter Driver

The USART is a hardware component that facilitates a connection between the mobile station and terminal equipment (e.g. a PC). This interface uses some of the circuits described in V.24.

The data exchange provided by this unit is serial and asynchronous (synchronous communication is not in the scope of this document). A driver that uses interrupts to manage a circular buffer for the sending and receiving direction is necessary in order to use this component in the GPRS. The driver has to be able to perform flow control.

2.8 TOM – Tunnelling of Messages

The TOM entity is present if and only if HS136 is supported (the feature flag FF_HS136 is enabled).

The main function of TOM is to tunnel non-GSM signalling messages between the MS and the SGSN. The only non-GSM signalling which is currently supported by TOM is for the EGPRS-136 system (according to TIA/EIA-136-376). Data transfer

in both uplink and downlink direction is possible. Two different priorities (high, low) of signalling data transfer are supported. TOM uses the unacknowledged mode of LLC and the acknowledged mode of GRR (RLC/MAC).

3 Parameters

/*

3.1 Primitive elements

*/
BYTE DTI_CHANNEL_TO_LOWER_LAYER 1
BYTE DTI_CHANNEL_TO_HIGHER_LAYER 0

LONG LINK_ID_PEER 25 /* DTI Link Id in peer direction */
LONG LINK_ID_PROT 300 /* DTI Link Id in protocol direction */

SHORTMSID_DUMMY 0 /* max slot identifier dummy*/

SHORTPPP_MRU_2000 2000 /* maximum receive unit 2000 */

LONG IP_ADDR 0xc6a80105 /* IP Address 198.168.1.2*/

LONG PPP_ACCM_0 0 /* Async-Control-Character-Map without mapping */

BYTE MSID_EXPECTED 15 /* expected msid in PPP_PDP_ACTIVATE_IND primitive */

LONG PPP_IP_141_64_21_2 0x8d401502 /* IP-Address 141.64.21.2 */
LONG PPP_PDNS_141_64_24_129 0x8d401881 /* primary DNS-Address 141.64.24.129 */
LONG PPP_SDNS_141_64_254_131 0x8d40fe83 /* secondary DNS-Address 141.64.254.131 */

LONG PPP_PDNS_DYNAMIC 0x0 /* dynamic primary DNS-Address */
LONG PPP_SDNS_DYNAMIC 0x0 /* dynamic secondary DNS-Address */

SHORT PPP_TERM_INSUF_RES 257 /* error cause from SNDCP */
/*

3.2 Struct elements

*/
BYTE LOGIN_NAME_LEN 8 /* length of login name */
BYTE LOGIN_PASSWORD_LEN 8 /* length of login password */
/*

3.3 Fields

*/
/*

3.3.1 LCP Configure Request

*/
/* HDLC Frame: LCP Configure Request (no options, Identifier = 1)

0xc8, 0x00	SDU length in bit (25 Byte = 200 Bit)
0x00, 0x00	SDU offset in bit

0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length (8 Byte)
0x07, 0x02	0x7d, 0x27, 0x7d, 0x22	PFC
0x08, 0x02	0x7d, 0x28, 0x7d, 0x22	ACFC
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest1)      0xc8, 0x00,
                             0x00, 0x00,
                             0x7e,
                             0xff,
                             0x7d, 0x23,
                             0xc0, 0x21,
                             0x7d, 0x21,
                             0x7d, 0x21,
                             0x7d, 0x20, 0x7d, 0x28,
                             0x7d, 0x27, 0x7d, 0x22,
                             0x7d, 0x28, 0x7d, 0x22,
                             0xf0, 0xb8,
                             0x7e
ENDFIELD(LCP_ConfRequest1, 29)
```

/* HDLC Frame: LCP Configure Request (no options, Identifier = 3)

0xc8, 0x00		SDU length in bit (25 Byte = 200 Bit)
0x00, 0x00		SDU offset in bit
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x03	0x7d, 0x23	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length (8 Byte)
0x07, 0x02	0x7d, 0x27, 0x7d, 0x22	PFC
0x08, 0x02	0x7d, 0x28, 0x7d, 0x22	ACFC
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest3)      0xc8, 0x00,
                             0x00, 0x00,
                             0x7e,
                             0xff,
                             0x7d, 0x23,
                             0xc0, 0x21,
                             0x7d, 0x21,
                             0x7d, 0x23,
```

```
                        0x7d, 0x20, 0x7d, 0x28,
                        0x7d, 0x27, 0x7d, 0x22,
                        0x7d, 0x28, 0x7d, 0x22,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfRequest3, 29)
```

/* HDLC Frame: LCP Configure Request (AuthProtocol = PAP, Identifier=4)

0xb8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x04	0x7d, 0x24	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest4)     0xb8, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0xc0, 0x21,
                        0x7d, 0x21,
                        0x7d, 0x24,
                        0x7d, 0x20, 0x7d, 0x28,
                        0x7d, 0x23,
                        0x7d, 0x24,
                        0xc0, 0x23,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfRequest4, 27)
```

/* HDLC Frame: LCP Configure Request (AuthProtocol = PAP, Address-and-Control-Field-Compression, Protocol-Field-Compression, Identifier=6)

0xf8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x06	0x7d, 0x26	Identifier
0x00, 0x0c	0x7d, 0x20, 0x7d, 0x2c	Length
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length

0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest6)      0xf8, 0x00,
                             0x00, 0x00,
                             0x7e,
                             0xff,
                             0x7d, 0x23,
                             0xc0, 0x21,
                             0x7d, 0x21,
                             0x7d, 0x26,
                             0x7d, 0x20, 0x7d, 0x2c,
                             0x7d, 0x23,
                             0x7d, 0x24,
                             0xc0, 0x23,
                             0x7d, 0x28,
                             0x7d, 0x22,
                             0x7d, 0x27,
                             0x7d, 0x22,
                             0xf0, 0xb8,
                             0x7e
ENDFIELD(LCP_ConfRequest6, 35)
```

/* HDLC Frame: LCP Configure Request (AuthProtocol = PAP, Address-and-Control-Field-Compression, Protocol-Field-Compression, Identifier=1, accm = 0, magic number = 0xa11cfc96)

0xa0, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x18	0x7d, 0x20, 0x7d, 0x38	Length
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d,0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x05	0x7d, 0x25	Option Type (magic number)
0x06	0x7d, 0x26	Option Length
0xa1, 0x1c, 0xfc, 0x96	0xa1, 0x7d, 0x3c, 0xfc, 0x96	Option Value (magic number)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequestMag) 0xa0, 0x01,
                          0x00, 0x00,
                          0x7e,
                          0xff,
                          0x7d, 0x23,
                          0xc0, 0x21,
                          0x7d, 0x21,
                          0x7d, 0x21,
                          0x7d, 0x20, 0x7d, 0x38,
                          0x7d, 0x22,
                          0x7d, 0x26,
                          0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                          0x7d, 0x23,
                          0x7d, 0x24,
                          0xc0, 0x23,
                          0x7d, 0x25,
                          0x7d, 0x26,
                          0xa1, 0x7d, 0x3c, 0xfc, 0x96,
                          0x7d, 0x27,
                          0x7d, 0x22,
                          0x7d, 0x28,
                          0x7d, 0x22,
                          0xf0, 0xb8,
                          0x7e
ENDFIELD(LCP_ConfRequestMag, 56)
```

/* HDLC Frame: LCP Configure Request (AuthProtocol = PAP, Address-and-Control-Field-Compression, Protocol-Field-Compression, Identifier=2, accm = 0)

0x58, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x22	Identifier
0x00, 0x12	0x7d, 0x20, 0x7d, 0x38	Length
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d,0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequestAccm)        0x58, 0x01,
                          0x00, 0x00,
                          0x7e,
                          0xff,
```

```
                0x7d, 0x23,
                0xc0, 0x21,
                0x7d, 0x21,
                0x7d, 0x22,
                0x7d, 0x20, 0x7d, 0x32,
                0x7d, 0x22,
                0x7d, 0x26,
                0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0x7d, 0x23,
                0x7d, 0x24,
                0xc0, 0x23,
                0x7d, 0x27,
                0x7d, 0x22,
                0x7d, 0x28,
                0x7d, 0x22,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_ConfRequestAccm, 47)

/* HDLC Frame: LCP Configure Request (AuthProtocol = CHAP, Address-and-Control-Field-
Compression, Protocol-Field-Compression, Identifier=1)
```

0x08, 0x01		SDU length (33 Byte = 264 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x0d	0x7d, 0x20, 0x7d, 0x2d	Length: 13
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ConfRequest50)
                0x08, 0x01,
                0x00, 0x00,
                0x7e,
                0xff, 0x7d, 0x23,
                0xc0, 0x21,
                0x7d, 0x21,
                0x7d, 0x21,
                0x7d, 0x20, 0x7d, 0x2d,
                0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
                0x7d, 0x27, 0x7d, 0x22,
                0x7d, 0x28, 0x7d, 0x22,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_ConfRequest50, 37)
```

/* HDLC Frame: LCP Configure Request (ACCM, Magic, ACFC, PFC, Callback, MMRRU, MED, Identifier=0)

0xf0, 0x02		SDU length (94 Byte = 752 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x00	0x7d, 0x20	Identifier
0x00, 0x32	0x7d, 0x20, 0x32	Length: 50
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x05	0x7d, 0x25	Option Type (Magic Number)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x54, 0x13	0x7d, 0x20, 0x7d, 0x20, 0x54, 0x7d, 0x33	Option Value
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0x0d	0x7d, 0x2d	Option Type (Callback)
0x03	0x7d, 0x23	Option Length
0x06	0x7d, 0x26	Option Value
0x11	0x7d, 0x31	Option Type (Multilink-MRRU)
0x04	0x7d, 0x24	Option Length
0x06, 0x4e	0x7d, 0x26, 4e	Option Value
0x13	0x7d, 0x33	Option Type (Multilink-Endpoint -Discriminator)
0x17	0x7d, 0x37	Option Length (23)
0x01, 0xad, 0xac, 0x80, 0xc0, 0x9b, 0x5e, 0x11, 0xd4, 0x94, 0x16, 0x00, 0x20, 0x18, 0x38, 0xd7, 0x67, 0x00, 0x00, 0x00, 0x00	0x7d, 0x21, 0xad, 0xac, 0x80, 0xc0, 0x9b, 0x5e, 0x7d, 0x31, 0xd4, 0x94, 0x7d, 0x36, 0x7d, 0x20, 0x20, 0x7d, 0x38, 0x38, 0xd7, 0x67, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest51)
                    0xf0, 0x02,
                    0x00, 0x00,
                    0x7e,
                    0xff, 0x7d, 0x23,
                    0xc0, 0x21,
```

```
                0x7d, 0x21,
                0x7d, 0x20,
                0x7d, 0x20, 0x32,
                0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0x7d, 0x25, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x54, 0x7d, 0x33,
                0x7d, 0x27, 0x7d, 0x22,
                0x7d, 0x28, 0x7d, 0x22,
                0x7d, 0x2d, 0x7d, 0x23, 0x7d, 0x26,
                0x7d, 0x31, 0x7d, 0x24, 0x7d, 0x26, 0x4e,
                0x7d, 0x33, 0x7d, 0x37, 0x7d, 0x21, 0xad, 0xac, 0x80, 0xc0, 0x9b, 0x5e,
                0x7d, 0x31, 0xd4, 0x94, 0x7d, 0x36, 0x7d, 0x20, 0x20, 0x7d, 0x38, 0x38,
                0xd7, 0x67, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_ConfRequest51, 98)
```

/* HDLC Frame: LCP Configure Request (ACCM, ACFC, PFC, Identifier=1)

0x28, 0x01		SDU length (37 Byte = 296 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x0e	0x7d, 0x20, 0x7d, 0x2e	Length: 14
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest53)
                0x28, 0x01,
                0x00, 0x00,
                0x7e,
                0xff, 0x7d, 0x23,
                0xc0, 0x21,
                0x7d, 0x21,
                0x7d, 0x21,
                0x7d, 0x20, 0x7d, 0x2e,
                0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0x7d, 0x27, 0x7d, 0x22,
                0x7d, 0x28, 0x7d, 0x22,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_ConfRequest53, 41)
```

/* HDLC Frame: Configure Request (LCP, PAP, ACFC, PFC, Identifier=1)

0xf8, 0x00	SDU length (31 Byte = 248 Bit)

0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x0c	0x7d, 0x20, 0x7d, 0x2c	Length: 12
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest66)
                        0xf8, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xff, 0x7d, 0x23,
                        0xc0, 0x21,
                        0x7d, 0x21,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x2c,
                        0x7d, 0x23, 0x7d, 0x24, 0xc0, 0x23,
                        0x7d, 0x27, 0x7d, 0x22,
                        0x7d, 0x28, 0x7d, 0x22,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfRequest66, 35)
```

/* HDLC Frame: Configure Request (LCP, ACFC, PFC, Identifier=1)

0xc8, 0x00		SDU length (25 Byte = 200 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length: 8
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest68)
        0xc8, 0x00,
        0x00, 0x00,
```

```
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x28,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest68, 29)

/* HDLC Frame: Configure Request (LCP, CHAP, ACFC, PFC, ACCM, Identifier=1)
```

0x68, 0x01		SDU length (45 Byte = 360 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x13	0x7d, 0x20, 0x7d, 0x33	Length: 19
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ConfRequest74)
        0x68, 0x01,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x33,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest74, 49)

/* HDLC Frame: Configure Request (LCP, PAP, ACFC, Identifier=2)
```

0xd8, 0x00		SDU length (27 Byte = 216 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x02	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest76)
        0xd8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x23, 0x7d, 0x24, 0xc0, 0x23,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest76, 31)
```

/* HDLC Frame: Configure Request (LCP, PAP, PFC, Identifier=2)

0xd8, 0x00		SDU length (27 Byte = 216 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x02	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x07	0x7d, 0x27	Option Type (PFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest78)
        0xd8, 0x00,
        0x00, 0x00,
        0x7e,
```

```
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x23, 0x7d, 0x24, 0xc0, 0x23,
        0x7d, 0x27, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest78, 31)
```

/* HDLC Frame: Configure Request (LCP, PAP, ACFC, PFC, Identifier=3)

0xf8, 0x00		SDU length (31 Byte = 248 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x03	0x7d, 0x23	Identifier
0x00, 0x0c	0x7d, 0x20, 0x7d, 0x2c	Length: 12
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest80)
        0xf8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x23,
        0x7d, 0x20, 0x7d, 0x2c,
        0x7d, 0x23, 0x7d, 0x24, 0xc0, 0x23,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest80, 35)
```

/* HDLC Frame: Configure Request (LCP, CHAP, ACFC, PFC, ACCM, MRU, Identifier=1)

0xa0, 0x01		SDU length (52 Byte = 416 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier

0x00, 0x17	0x7d, 0x20, 0x7d, 0x37	Length: 23
0x01	0x7d, 0x21	Option Type (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Option Value (2000)
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest82)
        0xa0, 0x01,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x37,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest82, 56)
```

/* HDLC Frame: Configure Request (LCP, CHAP, ACFC, PFC, ACCM, Identifier=2)

0x68, 0x01		SDU length (45 Byte = 360 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x02	0x7d, 0x22	Identifier
0x00, 0x13	0x7d, 0x20, 0x7d, 0x33	Length: 19
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)

0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest84)
        0x68, 0x01,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x33,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest84, 49)
```

/* HDLC Frame: Configure Request (LCP, CHAP, ACFC, PFC, MRU, Identifier=2)

0x40, 0x01		SDU length (40 Byte = 320 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x02	0x7d, 0x22	Identifier
0x00, 0x11	0x7d, 0x20, 0x7d, 0x31	Length: 17
0x01	0x7d, 0x21	Option Type (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Option Value (2000)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest86)
        0x40, 0x01,
        0x00, 0x00,
```

```
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x31,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest86, 44)

/* HDLC Frame: Configure Request (LCP, CHAP, ACFC, PFC, ACCM, MRU, Identifier=3)
```

0xa0, 0x01		SDU length (52 Byte = 416 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x03	0x7d, 0x23	Identifier
0x00, 0x17	0x7d, 0x20, 0x7d, 0x37	Length: 23
0x01	0x7d, 0x21	Option Type (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Option Value (2000)
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ConfRequest88)
        0xa0, 0x01,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x23,
        0x7d, 0x20, 0x7d, 0x37,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
```

```
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest88, 56)
```

/* HDLC Frame: Configure Request (LCP, ACFC, PFC, ACCM, MRU, Identifier=2)

0x60, 0x01		SDU length (44 Byte = 352 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x01	0x7d, 0x21	LCP Code (Configure Request)
0x02	0x7d, 0x22	Identifier
0x00, 0x12	0x7d, 0x20, 0x7d, 0x32	Length: 18
0x01	0x7d, 0x21	Option Type (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Option Value (2000)
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfRequest90)
        0x60, 0x01,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x21,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x32,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfRequest90, 48)
/*
```

3.3.2 LCP Configure Ack

*/
/* HDLC Frame: LCP Conf Ack (Identifier = 1)

0xc8, 0x00		SDU length in bit (25 Byte = 200 Bit)
0x00, 0x00		SDU offset in bit
0x7e		Flag
0xff	0xff	Address Field

0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length (8 Byte)
0x07, 0x02	0x7d, 0x27, 0x7d, 0x22	PFC
0x08, 0x02	0x7d, 0x28, 0x7d, 0x22	ACFC
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck1) 0xc8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xff,
                    0x7d, 0x23,
                    0xc0, 0x21,
                    0x7d, 0x22,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x28,
                    0x7d, 0x27, 0x7d, 0x22,
                    0x7d, 0x28, 0x7d, 0x22,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(LCP_ConfAck1, 29)
```

/* HDLC Frame: LCP Conf Ack (Identifier = 3)

0xc8, 0x00		SDU length in bit (25 Byte = 200 Bit)
0x00, 0x00		SDU offset in bit
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x03	0x7d, 0x23	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length (8 Byte)
0x07, 0x02	0x7d, 0x27, 0x7d, 0x22	PFC
0x08, 0x02	0x7d, 0x28, 0x7d, 0x22	ACFC
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck3) 0xc8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xff,
                    0x7d, 0x23,
                    0xc0, 0x21,
                    0x7d, 0x22,
                    0x7d, 0x23,
                    0x7d, 0x20, 0x7d, 0x28,
                    0x7d, 0x27, 0x7d, 0x22,
```

```
                        0x7d, 0x28, 0x7d, 0x22,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfAck3, 29)
```

/* HDLC Frame: LCP Configure Ack (Identifier = 4)

0xb8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x04	0x7d, 0x24	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck4) 0xb8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xff,
                    0x7d, 0x23,
                    0xc0, 0x21,
                    0x7d, 0x22,
                    0x7d, 0x24,
                    0x7d, 0x20, 0x7d, 0x28,
                    0x7d, 0x23,
                    0x7d, 0x24,
                    0xc0, 0x23,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(LCP_ConfAck4, 27)
```

/* HDLC Frame: LCP Configure Ack (AuthProtocol = PAP, Address-and-Control-Field-Compression, Protocol-Field-Compression ,Identifier=6)

0xf8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x06	0x7d, 0x26	Identifier
0x00, 0x0c	0x7d, 0x20, 0x7d, 0x2c	Length
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0x07	0x7d, 0x27	Option (PFC)

0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck6) 0xf8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xff,
                    0x7d, 0x23,
                    0xc0, 0x21,
                    0x7d, 0x22,
                    0x7d, 0x26,
                    0x7d, 0x20, 0x7d, 0x2c,
                    0x7d, 0x23,
                    0x7d, 0x24,
                    0xc0, 0x23,
                    0x7d, 0x28,
                    0x7d, 0x22,
                    0x7d, 0x27,
                    0x7d, 0x22,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(LCP_ConfAck6, 35)
```

/* HDLC Frame: LCP Configure Ack (AuthProtocol = PAP, Address-and-Control-Field-Compression, Protocol-Field-Compression, Identifier=2, accm = 0)

0x58, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Acknowledge)
0x01	0x7d, 0x22	Identifier
0x00, 0x12	0x7d, 0x20, 0x7d, 0x38	Length
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d,0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAccAccm)      0x58, 0x01,
                            0x00, 0x00,
                            0x7e,
                            0xff,
```

```
                0x7d, 0x23,
                0xc0, 0x21,
                0x7d, 0x22,
                0x7d, 0x22,
                0x7d, 0x20, 0x7d, 0x32,
                0x7d, 0x22,
                0x7d, 0x26,
                0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0x7d, 0x23,
                0x7d, 0x24,
                0xc0, 0x23,
                0x7d, 0x27,
                0x7d, 0x22,
                0x7d, 0x28,
                0x7d, 0x22,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_ConfAccAccm, 47)

/* HDLC Frame: LCP Configure Request (AuthProtocol = CHAP, Address-and-Control-Field-
Compression, Protocol-Field-Compression, Identifier=1)
```

0x08, 0x01		SDU length (33 Byte = 264 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x01	0x7d, 0x22	Identifier
0x00, 0x0d	0x7d, 0x20, 0x7d, 0x2d	Length: 13
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ConfAck50)
                0x08, 0x01,
                0x00, 0x00,
                0x7e,
                0xff, 0x7d, 0x23,
                0xc0, 0x21,
                0x7d, 0x22,
                0x7d, 0x21,
                0x7d, 0x20, 0x7d, 0x2d,
                0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
                0x7d, 0x27, 0x7d, 0x22,
                0x7d, 0x28, 0x7d, 0x22,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_ConfAck50, 37)
```

/* HDLC Frame: LCP Configure Ack (ACCM, ACFC, PFC, Identifier=1)

0x28, 0x01		SDU length (37 Byte = 296 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x0e	0x7d, 0x20, 0x7d, 0x2e	Length: 14
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck53)
                        0x28, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0xff, 0x7d, 0x23,
                        0xc0, 0x21,
                        0x7d, 0x22,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x2e,
                        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                        0x7d, 0x27, 0x7d, 0x22,
                        0x7d, 0x28, 0x7d, 0x22,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfAck53, 41)
```

/* HDLC Frame: Configure Ack (LCP, PAP, ACFC, PFC, Identifier=1)

0xf8, 0x00		SDU length (31 Byte = 248 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x0c	0x7d, 0x20, 0x7d, 0x2c	Length: 12
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length

0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck66)
                        0xf8, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xff, 0x7d, 0x23,
                        0xc0, 0x21,
                        0x7d, 0x22,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x2c,
                        0x7d, 0x23, 0x7d, 0x24, 0xc0, 0x23,
                        0x7d, 0x27, 0x7d, 0x22,
                        0x7d, 0x28, 0x7d, 0x22,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfAck66, 35)
```

/* HDLC Frame: Configure Ack (LCP, ACFC, PFC, Identifier=1)

0xc8, 0x00		SDU length (25 Byte = 200 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length: 8
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck68)
        0xc8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x22,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x28,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfAck68, 29)
```

/* HDLC Frame: Configure Ack (LCP, CHAP, ACFC, PFC, ACCM, Identifier=1)

0x68, 0x01	SDU length (45 Byte = 360 Bit)

0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x13	0x7d, 0x20, 0x7d, 0x33	Length: 19
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck74)
        0x68, 0x01,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x22,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x33,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfAck74, 49)
```

/* HDLC Frame: Configure Ack (LCP, PAP, ACFC, PFC, Identifier=3)

0xf8, 0x00		SDU length (31 Byte = 248 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x03	0x7d, 0x23	Identifier
0x00, 0x0c	0x7d, 0x20, 0x7d, 0x2c	Length: 12
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x04	0x7d, 0x24	Option Length
0xc0, 0x23	0xc0, 0x23	Option Value (PAP)

0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck80)
        0xf8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x22,
        0x7d, 0x23,
        0x7d, 0x20, 0x7d, 0x2c,
        0x7d, 0x23, 0x7d, 0x24, 0xc0, 0x23,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfAck80, 35)
```

/* HDLC Frame: Configure Ack (LCP, CHAP, ACFC, PFC, ACCM, MRU, Identifier=3)

0xa0, 0x01		SDU length (52 Byte = 416 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x02	0x7d, 0x22	LCP Code (Configure Ack)
0x03	0x7d, 0x23	Identifier
0x00, 0x17	0x7d, 0x20, 0x7d, 0x37	Length: 23
0x01	0x7d, 0x21	Option Type (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Option Value (2000)
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0x03	0x7d, 0x23	Option Type (Auth Proto)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23	0xc2, 0x23	Option Value (CHAP)
0x05	0x7d, 0x25	Algorithm MD5
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0x08	0x7d, 0x28	Option Type (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfAck88)
        0xa0, 0x01,
        0x00, 0x00,
```

```
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x22,
        0x7d, 0x23,
        0x7d, 0x20, 0x7d, 0x37,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0x7d, 0x27, 0x7d, 0x22,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfAck88, 56)
/*
```

3.3.3 LCP Configure Reject

```
*/
/* HDLC Frame: LCP Configure Reject (rejected option: magic number (packet:
LCP_ConfRequestMag))
```

0xd0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x05	0x7d, 0x25	Option Type (magic number)
0x06	0x7d, 0x26	Option Length
0xa1, 0x1c, 0xfc, 0x96	0xa1, 0x7d, 0x3c, 0xfc, 0x96	Option Value (magic number)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCPConfigureRejMag)
                        0xd0, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0xc0, 0x21,
                        0x7d, 0x24,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x2a,
                        0x7d, 0x25,
                        0x7d, 0x26,
                        0xa1, 0x7d, 0x3c, 0xfc, 0x96,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCPConfigureRejMag, 30)

/* HDLC Frame: LCP Configure Reject (Magic, Callback, MMRRU, MED, Identifier=0)
```

0x50, 0x02	SDU length (74 Byte = 592 Bit)
0x00, 0x00	SDU offset
0x7e	Flag

0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x00	0x7d, 0x20	Identifier
0x00, 0x28	0x7d, 0x20, 0x28	Length: 40
0x05	0x7d, 0x25	Option Type (Magic Number)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x54, 0x13	0x7d, 0x20, 0x7d, 0x20, 0x54, 0x7d, 0x33	Option Value
0x0d	0x7d, 0x2d	Option Type (Callback)
0x03	0x7d, 0x23	Option Length
0x06	0x7d, 0x26	Option Value
0x11	0x7d, 0x31	Option Type (Multilink-MRRU)
0x04	0x7d, 0x24	Option Length
0x06, 0x4e	0x7d, 0x26, 4e	Option Value
0x13	0x7d, 0x33	Option Type (Multilink-Endpoint -Discriminator)
0x17	0x7d, 0x37	Option Length (23)
0x01, 0xad, 0xac, 0x80, 0xc0, 0x9b, 0x5e, 0x11, 0xd4, 0x94, 0x16, 0x00, 0x20, 0x18, 0x38, 0xd7, 0x67, 0x00, 0x00, 0x00, 0x00	0x7d, 0x21, 0xad, 0xac, 0x80, 0xc0, 0x9b, 0x5e, 0x7d, 0x31, 0xd4, 0x94, 0x7d, 0x36, 0x7d, 0x20, 0x20, 0x7d, 0x38, 0x38, 0xd7, 0x67, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfReject51)
                        0x50, 0x02,
                        0x00, 0x00,
                        0x7e,
                        0xff, 0x7d, 0x23,
                        0xc0, 0x21,
                        0x7d, 0x24,
                        0x7d, 0x20,
                        0x7d, 0x20, 0x28,
                        0x7d, 0x25, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x54, 0x7d, 0x33,
                        0x7d, 0x2d, 0x7d, 0x23, 0x7d, 0x26,
                        0x7d, 0x31, 0x7d, 0x24, 0x7d, 0x26, 0x4e,
                        0x7d, 0x33, 0x7d, 0x37, 0x7d, 0x21, 0xad, 0xac, 0x80, 0xc0, 0x9b, 0x5e,
                        0x7d, 0x31, 0xd4, 0x94, 0x7d, 0x36, 0x7d, 0x20, 0x20, 0x7d, 0x38, 0x38,
                        0xd7, 0x67, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_ConfReject51, 78)
```

/* HDLC Frame: Configure Reject (LCP, PFC, Identifier=1)

0xa8, 0x00	SDU length (21 Byte = 168 Bit)

0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d, 0x26	Length: 6
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfReject66A)
        0xa8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x24,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x26,
        0x7d, 0x27, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfReject66A, 25)
```

/* HDLC Frame: Configure Reject (LCP, ACFC, Identifier=1)

0xa8, 0x00		SDU length (21 Byte = 168 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d, 0x26	Length: 6
0x08	0x7d, 0x28	Option (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfReject66B)
        0xa8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x24,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x26,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
```

ENDFIELD(LCP_ConfReject66B, 25)

/* HDLC Frame: Configure Reject (LCP, MRU, Identifier=1)

0xc0, 0x00		SDU length (24 Byte = 192 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length: 8
0x01	0x7d, 0x21	Option Type (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Option Value (2000)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfReject82A)
        0xc0, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x24,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x28,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfReject82A, 28)
```

/* HDLC Frame: Configure Reject (LCP, ACCM, Identifier=1)

0xe8, 0x00		SDU length (29 Byte = 232 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x02	0x7d, 0x22	Option Type (ACCM)
0x06	0x7d, 0x26	Option Length
0x0, 0x0, 0x0, 0x0	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (0)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfReject82B)
        0xe8, 0x00,
        0x00, 0x00,
        0x7e,
```

```
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x24,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfReject82B, 33)

/* HDLC Frame: Configure Reject (LCP, AP, Identifier=1)
```

0xc8, 0x00		SDU length (25 Byte = 200 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x04	0x7d, 0x24	LCP Code (Configure Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x09	0x7d, 0x20, 0x7d, 0x29	Length: 9
0x03	0x7d, 0x23	Option Type (AP)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23, 0x05	0xc2, 0x23, 0x7d, 0x25	Option Value (CHAP)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/

FIELD(LCP_ConfReject82C)
        0xc8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x24,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x29,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfReject82C, 29)
/*
```

3.3.4 LCP Configure NAK

```
*/

/* HDLC Frame: Configure NAK (LCP, PFC, Identifier=2)
```

0xa8, 0x00		SDU length (21 Byte = 168 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x03	0x7d, 0x23	LCP Code (Configure NAK)
0x02	0x7d, 0x22	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d,	Length: 6

	0x26	
0x07	0x7d, 0x27	Option (PFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfNAK76)
        0xa8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x23,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x26,
        0x7d, 0x27, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfNAK76, 25)
```

/* HDLC Frame: Configure NAK (LCP, ACFC, Identifier=2)

0xa8, 0x00		SDU length (21 Byte = 168 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x03	0x7d, 0x23	LCP Code (Configure NAK)
0x02	0x7d, 0x22	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d, 0x26	Length: 6
0x08	0x7d, 0x28	Option (ACFC)
0x02	0x7d, 0x22	Option Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfNAK78)
        0xa8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x23,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x26,
        0x7d, 0x28, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfNAK78, 25)
```

/* HDLC Frame: Configure NAK (LCP, MRU, Identifier=2)

0xc0, 0x00		SDU length (24 Byte = 192 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field

0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x03	0x7d, 0x23	LCP Code (Configure NAK)
0x02	0x7d, 0x22	Identifier
0x00, 0x08	0x7d, 0x20, 0x7d, 0x28	Length: 8
0x01	0x7d, 0x28	Option (MRU)
0x04	0x7d, 0x24	Option Length
0x07, 0xd0	0x7d, 0x27, 0xd0	Value (2000)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfNAK84)
        0xc0, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x23,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x28,
        0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x27, 0xd0,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfNAK84, 28)
```

/* HDLC Frame: Configure NAK (LCP, ACCM, Identifier=2)

0xe8, 0x00		SDU length (29 Byte = 232 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x03	0x7d, 0x23	LCP Code (Configure NAK)
0x02	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x02	0x7d, 0x28	Option (ACCM)
0x06	0x7d, 0x26	Option Length
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Value (0)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfNAK86)
        0xe8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x23,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0xf0, 0xb8,
        0x7e
```

ENDFIELD(LCP_ConfNAK86, 33)

/* HDLC Frame: Configure NAK (LCP, AP, Identifier=2)

0xc8, 0x00		SDU length (25 Byte = 200 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = (LCP)
0x03	0x7d, 0x23	LCP Code (Configure NAK)
0x02	0x7d, 0x22	Identifier
0x00, 0x09	0x7d, 0x20, 0x7d, 0x29	Length: 9
0x03	0x7d, 0x23	Option Type (AP)
0x05	0x7d, 0x25	Option Length
0xc2, 0x23, 0x05	0xc2, 0x23, 0x7d, 0x25	Option Value (CHAP)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ConfNAK90)
        0xc8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff, 0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x23,
        0x7d, 0x22,
        0x7d, 0x20, 0x7d, 0x29,
        0x7d, 0x23, 0x7d, 0x25, 0xc2, 0x23, 0x7d, 0x25,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ConfNAK90, 29)
/*
```

3.3.5 LCP Terminate Request

*/

/* HDLC Frame: Terminate Request (LCP, Identifier = 1, Error-Code = PPP_TERM_OK_MMI)

0xa0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x05	0x7d, 0x25	LCP Code (Terminate Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d, 0x26	Length
0xcf, 0x01	0xcf, 0x7d, 0x21	Internal Error Code (PPP_TERM_OK_MMI)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_TermReq0)
        0xa0, 0x00,
        0x00, 0x00,
```

```
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x25,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x26,
        0xcf, 0x7d, 0x21,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_TermReq0, 24)

/* HDLC Frame: Terminate Request (LCP, Identifier = 1, Error-Code = PPP_TERM_INSUF_RES)
```

0xa8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x05	0x7d, 0x25	LCP Code (Terminate Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d, 0x26	Length
0x01, 0x01	0x7d, 0x21, 0x7d, 0x21	Internal Error Code (PPP_TERM_INSUF_RES)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_TermRequest90)
        0xa8, 0x00,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x25,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x26,
        0x7d, 0x21, 0x7d, 0x21,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_TermRequest90, 25)

/*
```

3.3.6 LCP Terminate Ack

```
*/
/* HDLC Frame: Terminate Ack (LCP, Identifier = 1, Error-Code = PPP_TERM_OK_MMI)
```

0xa0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x06	0x7d, 0x26	LCP Code (Terminate Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d,	Length

	0x26	
0xcf, 0x01	0xcf, 0x7d, 0x21	Internal Error Code (== t_cause)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(LCP_TermAck0)
0xa0, 0x00,
0x00, 0x00,
0x7e,
0xff,
0x7d, 0x23,
0xc0, 0x21,
0x7d, 0x26,
0x7d, 0x21,
0x7d, 0x20, 0x7d, 0x26,
0xcf, 0x7d, 0x21,
0xf0, 0xb8,
0x7e
ENDFIELD(LCP_TermAck0, 24)

/* HDLC Frame: Terminate Ack (LCP, Identifier = 1, Error-Code = PPP_TERM_INSUF_RES)

0xa8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x06	0x7d, 0x26	LCP Code (Terminate Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x06	0x7d, 0x20, 0x7d, 0x26	Length
0x01, 0x01	0x7d, 0x21, 0x7d, 0x21	Internal Error Code (PPP_TERM_INSUF_RES)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(LCP_TermAck90)
0xa8, 0x00,
0x00, 0x00,
0x7e,
0xff,
0x7d, 0x23,
0xc0, 0x21,
0x7d, 0x26,
0x7d, 0x21,
0x7d, 0x20, 0x7d, 0x26,
0x7d, 0x21, 0x7d, 0x21,
0xf0, 0xb8,
0x7e
ENDFIELD(LCP_TermAck90, 25)

/*

3.3.7 LCP Code Reject

*/
/* HDLC Frame: Code Reject (LCP, rejection of Identification55, identifier=1)

0x58, 0x01	SDU length (43 Bytes = 344 Bit)

0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x07	0x7d, 0x27	LCP Code (Code Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x0c	0x7d, 0x2c	rejected LCP Code (Identification)
0x02	0x7d, 0x22	Identifier
0x00, 0x12	0x7d, 0x20, 0x7d, 0x32	Length: 18
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Magic Number
0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30	0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30	Information "MSRASV4.00"
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(LCP_CodeReject55)
0x58, 0x01,
0x00, 0x00,
0x7e,
0xff, 0x7d, 0x23,
0xc0, 0x21,
0x7d, 0x27,
0x7d, 0x21,
0x7d, 0x20, 0x7d, 0x36,
0x7d, 0x2c,
0x7d, 0x22,
0x7d, 0x20, 0x7d, 0x32,
0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30,
0xf0, 0xb8,
0x7e
ENDFIELD(LCP_CodeReject55, 47)

/* HDLC Frame: Code Reject (LCP, rejection of Identification57, identifier=2)

0x60, 0x01		SDU length (44 Byte = 352 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x07	0x7d, 0x27	LCP Code (Identification)
0x02	0x7d, 0x22	Identifier
0x00, 0x17	0x7d, 0x20, 0x7d, 0x37	Length: 23
0x0c	0x7d, 0x2c	rejected LCP Code (Identification)
0x03	0x7d, 0x23	Identifier
0x00, 0x13	0x7d, 0x20, 0x7d,	Length: 19

	0x33	
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Magic Number
0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57	0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57	Information "MSRAS-1-STW"
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(LCP_CodeReject57)
0x60, 0x01,
0x00, 0x00,
0x7e,
0xff, 0x7d, 0x23,
0xc0, 0x21,
0x7d, 0x27,
0x7d, 0x22,
0x7d, 0x20, 0x7d, 0x37,
0x7d, 0x2c,
0x7d, 0x23,
0x7d, 0x20, 0x7d, 0x33,
0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57,
0xf0, 0xb8,
0x7e
ENDFIELD(LCP_CodeReject57, 48)

/*

3.3.8 LCP Protocol Reject

*/
/* HDLC Frame: LCP Protocol Reject (Identifier = 1, rejected packet is CCPConfReq0)

0x38, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x10	0x7d, 0x20, 0x7d, 0x30	Length
0x80, 0xfd	0x80, 0xfd	Protocol = CCP
0x01	0x7d, 0x21	CCP Code (Conf Req)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (OUI)
0x06	0x7d, 0x26	Option Length
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (dummy)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCPProtoRej0)  0x38, 0x01,
                     0x00, 0x00,
                     0x7e,
                     0xff,
                     0x7d, 0x23,
                     0xc0, 0x21,
                     0x7d, 0x28,
                     0x7d, 0x21,
                     0x7d, 0x20, 0x7d, 0x30,
                     0x80, 0xfd,
                     0x7d, 0x21,
                     0x7d, 0x21,
                     0x7d, 0x20, 0x7d, 0x2a,
                     0x7d, 0x23,
                     0x7d, 0x26,
                     0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                     0xf0, 0xb8,
                     0x7e
ENDFIELD(LCPProtoRej0, 43)
```

/* HDLC Frame: LCP Protocol Reject packet (rejected packet is INVALID_FRAME2)

0x30, 0x04		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x77	0x7d, 0x20, 0x77	Length
0x00	0x7d, 0x20	Rejected packet:
0xf7	0xf7	Address Field (invalid!!!)
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	110 byte data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCPProtoRej1)  0x30, 0x04,
                     0x00, 0x00,
                     0x7e,
                     0xff,
                     0x7d, 0x23,
                     0xc0, 0x21,
                     0x7d, 0x28,
                     0x7d, 0x21,
                     0x7d, 0x20, 0x77,
                     0x7d, 0x20,
                     0xf7,
                     0x7d, 0x23,
                     0x7d, 0x020, 0x21,
                     0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                     0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                     0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                     0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                     0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
```

```
                  0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                  0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                  0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                  0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                  0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                  0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                  0xf0, 0xb8,
                  0x7e
ENDFIELD(LCPProtoRej1, 138)

/* HDLC Frame: Protocol Reject (LCP, Identifier = 1)
```

0x88, 0x0c		SDU length (401 Byte = 3208 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0xdb	0x7d, 0x20, 0xdb	Length: 219
0x80, 0xfd	0x80, 0xfd	Rejected Protocol (CCP)
…	…	Rejected Packet Content
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ProtoReject59)
                  0x88, 0x0c,
                  0x00, 0x00,
                  0x7e,
                  0xff,
                  0x7d, 0x23,
                  0xc0, 0x21,
                  0x7d, 0x28,
                  0x7d, 0x21,
                  0x7d, 0x20, 0xdb,
                  0x80, 0xfd,
                  0x7d, 0x21,
                  0x7d, 0x24,
                  0x7d, 0x20, 0xd5,
                  0xfe, 0xcb, 0x7d, 0x21, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                  0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                  0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x74, 0xff,
                  0xc9, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x68,
                  0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0xa0, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x30, 0x30, 0x20, 0x62, 0x7d, 0x25, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0xc0, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0xa4, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0xbc, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x38,
                  0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x70, 0x7d, 0x20, 0xea, 0x7d, 0x20,
                  0xb0, 0xc7, 0x7d, 0x34, 0x7d, 0x20, 0x69, 0xcc, 0xf1, 0x77, 0x7d, 0x20,
                  0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x20, 0x7d, 0x20,
                  0x7d, 0x20, 0xe8, 0xfe, 0xc9, 0x7d, 0x20, 0xc8, 0xcd, 0xf1, 0x77, 0x7d,
                  0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                  0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
```

```
0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x74, 0x7d, 0x20, 0x6e, 0x7d,
0x20, 0x7d, 0x3d, 0x7d, 0x20, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21,
0x7d, 0x2b, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x7d, 0x20, 0xdb, 0x80, 0xfd, 0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x20, 0xd5,
0xfe, 0xcb, 0x7d, 0x21, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x74, 0xff,
0xc9, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x68,
0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x32, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20,
0x7d, 0x20, 0x7d, 0x26,
0xf0, 0xb8,
0x7e
ENDFIELD(LCP_ProtoReject59, 405)
```

/* HDLC Frame: Protocol Reject (LCP, rejected packet is INVALID_FRAME2, Identifier = 2)

0x30, 0x04		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x02	0x7d, 0x22	Identifier
0x00, 0x77	0x7d, 0x20, 0x77	Length
0x00	0x7d, 0x20	Rejected packet:
0xf7	0xf7	Address Field (invalid!!!)
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	110 byte data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ProtoReject88)
        0x30, 0x04,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x28,
        0x7d, 0x22,
        0x7d, 0x20, 0x77,
        0x7d, 0x20,
        0xf7,
        0x7d, 0x23,
        0x7d, 0x020, 0x21,
        0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
```

```
        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ProtoReject88, 138)

/* HDLC Frame: Protocol Reject (LCP, rejected protocol = uncompressed TCP, identifier = 1)
```

0x10, 0x06		SDU length (194 Byte = 1552 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x01	0x7d, 0x21	Identifier
0x00, 0x9a	0x7d, 0x20, 0x9a	Length: 154
0x00, 0x2f	0x7d, 0x20, 0x2f	Rejected Protocol (UTCP)
...	...	Rejected Packet Content
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ProtoReject_D0)
        0x10, 0x06,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x28,
        0x7d, 0x21,
        0x7d, 0x20, 0x9a,
        0x7d, 0x20, 0x2f,
        0x45, 0x7d, 0x20, 0x7d, 0x20, 0x94, 0x7d, 0x24, 0x7d, 0x3e, 0x40, 0x7d, 0x20, 0x75, 0x7d,
0x20,
        0xd9, 0x26, 0x3e, 0x60, 0xcf, 0x7d, 0x29, 0x97, 0xbd, 0x82, 0xf8, 0x7d, 0x20, 0x50, 0x7d,
0x24,
        0x7d, 0x30, 0xe2, 0x75, 0xdd, 0x43, 0x7d, 0x20, 0x7d, 0x3a, 0xbe, 0x78, 0x50, 0x7d, 0x38,
0x21,
        0x7d, 0x29, 0x7d, 0x38, 0xaf, 0x7d, 0x20, 0x7d, 0x20, 0x48, 0x54, 0x54, 0x50, 0x2f, 0x31,
0x2e,
        0x31, 0x20, 0x32, 0x30, 0x30, 0x20, 0x4f, 0x4b, 0x7d, 0x2d, 0x7d, 0x2a, 0x53, 0x65, 0x72,
0x76,
        0x65, 0x72, 0x3a, 0x20, 0x4d, 0x69, 0x63, 0x72, 0x6f, 0x73, 0x6f, 0x66, 0x74, 0x2d, 0x49,
0x49,
        0x53, 0x2f, 0x34, 0x2e, 0x30, 0x7d, 0x2d, 0x7d, 0x2a, 0x44, 0x61, 0x74, 0x65, 0x3a, 0x20,
0x46,
        0x72, 0x69, 0x2c, 0x20, 0x30, 0x36, 0x20, 0x4f, 0x63, 0x74, 0x20, 0x32, 0x30, 0x30, 0x30,
0x20,
        0x31, 0x32, 0x3a, 0x34, 0x35, 0x3a, 0x32, 0x38, 0x20, 0x47, 0x4d, 0x54, 0x7d, 0x2d, 0x7d,
0x2a,
        0x43, 0x6f, 0x6e, 0x74, 0x65, 0x6e, 0x74, 0x2d, 0x54, 0x79, 0x70, 0x65, 0x3a, 0x20, 0x74,
0x65,
        0x78, 0x74, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ProtoReject_D0, 198)
```

/* HDLC Frame: Protocol Reject (LCP, rejected protocol = uncompressed TCP, identifier = 2)

0x10, 0x06		SDU length (194 Byte = 1552 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x02	0x7d, 0x22	Identifier
0x00, 0x9a	0x7d, 0x20, 0x9a	Length: 154
0x00, 0x2f	0x7d, 0x20, 0x2f	Rejected Protocol (UTCP)
…	…	Rejected Packet Content
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

*/

```
FIELD(LCP_ProtoReject_D0B)
        0x10, 0x06,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x28,
        0x7d, 0x22,
        0x7d, 0x20, 0x9a,
        0x7d, 0x20, 0x2f,
        0x45, 0x7d, 0x20, 0x7d, 0x20, 0x94, 0x7d, 0x24, 0x7d, 0x3e, 0x40, 0x7d, 0x20, 0x75, 0x7d,
0x20,
        0xd9, 0x26, 0x3e, 0x60, 0xcf, 0x7d, 0x29, 0x97, 0xbd, 0x82, 0xf8, 0x7d, 0x20, 0x50, 0x7d,
0x24,
        0x7d, 0x30, 0xe2, 0x75, 0xdd, 0x43, 0x7d, 0x20, 0x7d, 0x3a, 0xbe, 0x78, 0x50, 0x7d, 0x38,
0x21,
        0x7d, 0x29, 0x7d, 0x38, 0xaf, 0x7d, 0x20, 0x7d, 0x20, 0x48, 0x54, 0x54, 0x50, 0x2f, 0x31,
0x2e,
        0x31, 0x20, 0x32, 0x30, 0x30, 0x20, 0x4f, 0x4b, 0x7d, 0x2d, 0x7d, 0x2a, 0x53, 0x65, 0x72,
0x76,
        0x65, 0x72, 0x3a, 0x20, 0x4d, 0x69, 0x63, 0x72, 0x6f, 0x73, 0x6f, 0x66, 0x74, 0x2d, 0x49,
0x49,
        0x53, 0x2f, 0x34, 0x2e, 0x30, 0x7d, 0x2d, 0x7d, 0x2a, 0x44, 0x61, 0x74, 0x65, 0x3a, 0x20,
0x46,
        0x72, 0x69, 0x2c, 0x20, 0x30, 0x36, 0x20, 0x4f, 0x63, 0x74, 0x20, 0x32, 0x30, 0x30, 0x30,
0x20,
        0x31, 0x32, 0x3a, 0x34, 0x35, 0x3a, 0x32, 0x38, 0x20, 0x47, 0x4d, 0x54, 0x7d, 0x2d, 0x7d,
0x2a,
        0x43, 0x6f, 0x6e, 0x74, 0x65, 0x6e, 0x74, 0x2d, 0x54, 0x79, 0x70, 0x65, 0x3a, 0x20, 0x74,
0x65,
        0x78, 0x74, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ProtoReject_D0B, 198)
```

/* HDLC Frame: Protocol Reject (LCP, rejected protocol = compressed TCP, identifier = 2)

0x98, 0x0e		SDU length (467 Byte = 3736 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field

0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x02	0x7d, 0x22	Identifier
0x01, 0xa5	0x7d, 0x21, 0xa5	Length: 421
0x00, 0x2d	0x7d, 0x20, 0x2d	Rejected Protocol (CTCP)
...	...	Rejected Packet Content
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ProtoReject_E0)
        0x98, 0x0e,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x28,
        0x7d, 0x22,
        0x7d, 0x21, 0xa5,
        0x7d, 0x20, 0x2d,
        0x3f, 0xbf, 0x67, 0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x20, 0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e,
0x7d,
        0x2d, 0x7d, 0x2a, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d,
0x7d,
        0x2a, 0x3c, 0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x43, 0x6f, 0x6e, 0x64, 0x61, 0x74, 0x3c,
0x2f,
        0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d,
0x2d,
        0x7d, 0x2a, 0x3c, 0x73, 0x63, 0x72, 0x69, 0x70, 0x74, 0x20, 0x6c, 0x61, 0x6e, 0x67, 0x75,
0x61,
        0x67, 0x65, 0x3d, 0x22, 0x4a, 0x61, 0x76, 0x61, 0x53, 0x63, 0x72, 0x69, 0x70, 0x74, 0x22,
0x3e,
        0x20, 0x3c, 0x21, 0x2d, 0x2d, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d,
0x7d,
        0x2a, 0x66, 0x75, 0x6e, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x20, 0x4d, 0x4d, 0x5f, 0x73, 0x77,
0x61,
        0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x28, 0x29, 0x20, 0x7b, 0x20, 0x2f, 0x2f, 0x76, 0x31,
0x2e,
        0x30, 0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x76, 0x61, 0x72, 0x20, 0x69, 0x2c, 0x74, 0x68,
0x65,
        0x4f, 0x62, 0x6a, 0x2c, 0x6a, 0x3d, 0x30, 0x2c, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72,
0x61,
        0x79, 0x3d, 0x6e, 0x65, 0x77, 0x20, 0x41, 0x72, 0x72, 0x61, 0x79, 0x2c, 0x6f, 0x6c, 0x64,
0x41,
        0x72, 0x72, 0x61, 0x79, 0x3d, 0x64, 0x6f, 0x63, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x2e, 0x4d,
0x4d,
        0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x67, 0x44, 0x61, 0x74, 0x61, 0x3b, 0x7d, 0x2d,
0x7d,
        0x2a, 0x20, 0x20, 0x66, 0x6f, 0x72, 0x20, 0x28, 0x69, 0x3d, 0x30, 0x3b, 0x20, 0x69, 0x20,
0x3c,
        0x20, 0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e,
0x61,
        0x72, 0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x2e, 0x6c, 0x65, 0x6e, 0x67, 0x74, 0x68,
0x2d,
        0x32, 0x29, 0x3b, 0x20, 0x69, 0x2b, 0x3d, 0x33, 0x29, 0x20, 0x7b, 0x7d, 0x2d, 0x7d, 0x2a,
0x20,
        0x20, 0x20, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a, 0x20, 0x3d, 0x20, 0x65, 0x76, 0x61,
0x6c,
```

```
        0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e, 0x61,
0x72,
        0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x5b, 0x28, 0x6e, 0x61, 0x76, 0x69, 0x67, 0x61,
0x74,
        0x6f, 0x72, 0x2e, 0x61, 0x70, 0x70, 0x4e, 0x61, 0x6d, 0x65, 0x20, 0x3d, 0x3d, 0x20, 0x27,
0x4e,
        0x65, 0x74, 0x73, 0x63, 0x61, 0x70, 0x65, 0x27, 0x29, 0x3f, 0x69, 0x3a, 0x69, 0x2b, 0x31,
0x5d,
        0x29, 0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x69, 0x66, 0x20, 0x28, 0x74, 0x68,
0x65,
        0x4f, 0x62, 0x6a, 0x20, 0x21, 0x3d, 0x20, 0x6e, 0x75, 0x6c, 0x6c, 0x29, 0x20, 0x7b, 0x7d,
0x2d,
        0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72,
0x61,
        0x79, 0x5b, 0x6a, 0x2b, 0x2b, 0x5d, 0x20, 0x3d, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a,
0x3b,
        0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41,
0x72,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ProtoReject_E0, 471)

/* HDLC Frame: Protocol Reject (LCP, rejected protocol = compressed TCP, identifier = 3)
```

0x98, 0x0e		SDU length (467 Byte = 3736 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x08	0x7d, 0x28	LCP Code (Protocol Reject)
0x03	0x7d, 0x23	Identifier
0x01, 0xa5	0x7d, 0x21, 0xa5	Length: 421
0x00, 0x2d	0x7d, 0x20, 0x2d	Rejected Protocol (CTCP)
...	...	Rejected Packet Content
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

```
*/
FIELD(LCP_ProtoReject_E0B)
        0x98, 0x0e,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0xc0, 0x21,
        0x7d, 0x28,
        0x7d, 0x23,
        0x7d, 0x21, 0xa5,
        0x7d, 0x20, 0x2d,
        0x3f, 0xbf, 0x67, 0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x20, 0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e,
0x7d,
        0x2d, 0x7d, 0x2a, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d,
0x7d,
        0x2a, 0x3c, 0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x43, 0x6f, 0x6e, 0x64, 0x61, 0x74, 0x3c,
0x2f,
        0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d,
0x2d,
        0x7d, 0x2a, 0x3c, 0x73, 0x63, 0x72, 0x69, 0x70, 0x74, 0x20, 0x6c, 0x61, 0x6e, 0x67, 0x75,
0x61,
```

```
        0x67, 0x65, 0x3d, 0x22, 0x4a, 0x61, 0x76, 0x61, 0x53, 0x63, 0x72, 0x69, 0x70, 0x74, 0x22,
0x3e,
        0x20, 0x3c, 0x21, 0x2d, 0x2d, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d,
0x7d,
        0x2a, 0x66, 0x75, 0x6e, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x20, 0x4d, 0x4d, 0x5f, 0x73, 0x77,
0x61,
        0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x28, 0x29, 0x20, 0x7b, 0x20, 0x2f, 0x2f, 0x76, 0x31,
0x2e,
        0x30, 0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x76, 0x61, 0x72, 0x20, 0x69, 0x2c, 0x74, 0x68,
0x65,
        0x4f, 0x62, 0x6a, 0x2c, 0x6a, 0x3d, 0x30, 0x2c, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72,
0x61,
        0x79, 0x3d, 0x6e, 0x65, 0x77, 0x20, 0x41, 0x72, 0x72, 0x61, 0x79, 0x2c, 0x6f, 0x6c, 0x64,
0x41,
        0x72, 0x72, 0x61, 0x79, 0x3d, 0x64, 0x6f, 0x63, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x2e, 0x4d,
0x4d,
        0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x67, 0x44, 0x61, 0x74, 0x61, 0x3b, 0x7d, 0x2d,
0x7d,
        0x2a, 0x20, 0x20, 0x66, 0x6f, 0x72, 0x20, 0x28, 0x69, 0x3d, 0x30, 0x3b, 0x20, 0x69, 0x20,
0x3c,
        0x20, 0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e,
0x61,
        0x72, 0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x2e, 0x6c, 0x65, 0x6e, 0x67, 0x74, 0x68,
0x2d,
        0x32, 0x29, 0x3b, 0x20, 0x69, 0x2b, 0x3d, 0x33, 0x29, 0x20, 0x7b, 0x7d, 0x2d, 0x7d, 0x2a,
0x20,
        0x20, 0x20, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a, 0x20, 0x3d, 0x20, 0x65, 0x76, 0x61,
0x6c,
        0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e, 0x61,
0x72,
        0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x5b, 0x28, 0x6e, 0x61, 0x76, 0x69, 0x67, 0x61,
0x74,
        0x6f, 0x72, 0x2e, 0x61, 0x70, 0x70, 0x4e, 0x61, 0x6d, 0x65, 0x20, 0x3d, 0x3d, 0x20, 0x27,
0x4e,
        0x65, 0x74, 0x73, 0x63, 0x61, 0x70, 0x65, 0x27, 0x29, 0x3f, 0x69, 0x3a, 0x69, 0x2b, 0x31,
0x5d,
        0x29, 0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x69, 0x66, 0x20, 0x28, 0x74, 0x68,
0x65,
        0x4f, 0x62, 0x6a, 0x20, 0x21, 0x3d, 0x20, 0x6e, 0x75, 0x6c, 0x6c, 0x29, 0x20, 0x7b, 0x7d,
0x2d,
        0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72,
0x61,
        0x79, 0x5b, 0x6a, 0x2b, 0x2b, 0x5d, 0x20, 0x3d, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a,
0x3b,
        0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41,
0x72,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_ProtoReject_E0B, 471)

/*
```

3.3.9 LCP Identification

```
*/
/* HDLC Frame: Identification (LCP, ACFC, PFC, Identifier=2)
```

0x00, 0x01	SDU length (32 Bytes = 256 Bit)
0x00, 0x00	SDU offset
0x7e	Flag

0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x0c	0x7d, 0x2c	LCP Code (Identification)
0x02	0x7d, 0x22	Identifier
0x00, 0x12	0x7d, 0x20, 0x7d, 0x32	Length: 18
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Magic Number
0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30	0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30	Information "MSRASV4.00"
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_Identification55)
                        0x00, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0xc0, 0x21,
                        0x7d, 0x2c,
                        0x7d, 0x22,
                        0x7d, 0x20, 0x7d, 0x32,
                        0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                        0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(LCP_Identification55, 36)

/* HDLC Frame: Identification (LCP, ACFC, PFC, Identifier=3)
```

0x08, 0x01		SDU length (33 Byte = 264 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xc0, 0x21	0xc0, 0x21	Protocol = LCP
0x0c	0x7d, 0x2c	LCP Code (Identification)
0x03	0x7d, 0x23	Identifier
0x00, 0x13	0x7d, 0x20, 0x7d, 0x33	Length: 19
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Magic Number
0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57	0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57	Information "MSRAS-1-STW"
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(LCP_Identification57)
                        0x08, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0xc0, 0x21,
```

```
                0x7d, 0x2c,
                0x7d, 0x23,
                0x7d, 0x20, 0x7d, 0x33,
                0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57,
                0xf0, 0xb8,
                0x7e
ENDFIELD(LCP_Identification57, 37)

/* HDLC Frame: Identification (LCP, ACFC, PFC, ACCM=0, Identifier=2)
```

0xc0, 0x00	SDU length (24 Bytes = 192 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0xc0, 0x21	Protocol = LCP
0x0c	LCP Code (Identification)
0x02	Identifier
0x00, 0x12	Length: 18
0x00, 0x00, 0x00, 0x00	Magic Number
0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30	Information "MSRASV4.00"
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
*/
FIELD(LCP_Identification77)
        0xc0, 0x00,
        0x00, 0x00,
        0x7e,
        0xc0, 0x21,
        0x0c,
        0x02,
        0x00, 0x12,
        0x00, 0x00, 0x00, 0x00,
        0x4d, 0x53, 0x52, 0x41, 0x53, 0x56, 0x34, 0x2e, 0x30, 0x30,
        0xf0, 0xb8,
        0x7e
ENDFIELD(LCP_Identification77, 28)

/* HDLC Frame: Identification (LCP, ACFC, PFC, ACCM=0, Identifier=3)
```

0xc8, 0x00	SDU length (25 Byte = 200 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0xc0, 0x21	Protocol = LCP
0x0c	LCP Code (Identification)
0x03	Identifier
0x00, 0x13	Length: 19
0x00, 0x00, 0x00, 0x00	Magic Number
0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57	Information "MSRAS-1-STW"
0xf0, 0xb8	FSC (dummy)

0x7e	Flag

*/
FIELD(LCP_Identification79)
0xc8, 0x00,
0x00, 0x00,
0x7e,
0xc0, 0x21,
0x0c,
0x03,
0x00, 0x13,
0x00, 0x00, 0x00, 0x00,
0x4d, 0x53, 0x52, 0x41, 0x53, 0x2d, 0x31, 0x2d, 0x53, 0x54, 0x57,
0xf0, 0xb8,
0x7e
ENDFIELD(LCP_Identification79, 29)

/*

3.3.10 CHAP Challenge

*/
/* HDLC Frame: Challenge (CHAP, ACFC, PFC, Identifier = 1)

0x80, 0x01		SDU length (48 Byte = 384 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xc2, 0x23	0xc2, 0x23	Protocol = CHAP
0x01	0x7d, 0x21	CHAP Code (Challenge)
0x01	0x7d, 0x21	Identifier
0x00, 0x15	0x7d, 0x20, 0x7d, 0x35	Length: 21
0x10	0x7d, 0x30	Value-Size: 16
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f	0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x22, 0x7d, 0x23, 0x7d, 0x24, 0x7d, 0x25, 0x7d, 0x26, 0x7d, 0x27, 0x7d, 0x28, 0x7d, 0x29, 0x7d, 0x2a, 0x7d, 0x2b, 0x7d, 0x2c, 0x7d, 0x2d, 0x7d, 0x2e, 0x7d, 0x2f	Value
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

*/
FIELD(CHAP_Challenge54)
0x80, 0x01,
0x00, 0x00,
0x7e,
0xc2, 0x23,
0x7d, 0x21,
0x7d, 0x21,
0x7d, 0x20, 0x7d, 0x35,
0x7d, 0x30,

```
                0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x22, 0x7d, 0x23, 0x7d, 0x24, 0x7d, 0x25,
                0x7d, 0x26, 0x7d, 0x27, 0x7d, 0x28, 0x7d, 0x29, 0x7d, 0x2a, 0x7d, 0x2b,
                0x7d, 0x2c, 0x7d, 0x2d, 0x7d, 0x2e, 0x7d, 0x2f,
                0xf0, 0xb8,
                0x7e
ENDFIELD(CHAP_Challenge54, 52)

/* HDLC Frame: Challenge  (CHAP, ACFC, PFC, ACCM=0, Identifier = 1)
```

0xd8, 0x00	SDU length (27 Byte = 216 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0xc2, 0x23	Protocol = CHAP
0x01	CHAP Code (Challenge)
0x01	Identifier
0x00, 0x15	Length: 21
0x10	Value-Size: 16
0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f	Value
0xf0, 0xb8	FCS (dummy)
0x7e	Flag

```
*/
FIELD(CHAP_Challenge76)
        0xd8, 0x00,
        0x00, 0x00,
        0x7e,
        0xc2, 0x23,
        0x01,
        0x01,
        0x00, 0x15,
        0x10,
        0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e,
0x0f,
        0xf0, 0xb8,
        0x7e
ENDFIELD(CHAP_Challenge76, 31)

/*
```

3.3.11 CHAP Response

```
*/
/* HDLC Frame: Response  (CHAP, ACFC, PFC, Identifier = 1)
```

0x68, 0x01		SDU length (45 Byte = 360 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xc2, 0x23	0xc2, 0x23	Protocol = CHAP
0x02	0x7d, 0x22	CHAP Code (Response)
0x01	0x7d, 0x21	Identifier
0x00, 0x1d	0x7d, 0x20, 0x7d, 0x3d	Length: 29
0x10	0x7d, 0x30	Value-Size: 16
0xd1, 0xe3,	0xd1, 0xe3,	Value

0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3, 0x13, 0x08, 0xc2, 0x2f, 0xd7	0x7f, 0x7d, 0x22, 0x2a, 0x7d, 0x2c, 0xcd, 0x7d, 0x33, 0x85, 0xd4, 0xe3, 0x7d, 0x33, 0x7d, 0x28, 0xc2, 0x2f, 0xd7	
0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	Name "testname"
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

*/

```
FIELD(CHAP_Response54)
                        0x68, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0xc2, 0x23,
                        0x7d, 0x22,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x3d,
                        0x7d, 0x30,
                        0xd1, 0xe3, 0x7f, 0x7d, 0x22, 0x2a, 0x7d, 0x2c, 0xcd, 0x7d, 0x33, 0x85,
                        0xd4, 0xe3, 0x7d, 0x33, 0x7d, 0x28, 0xc2, 0x2f, 0xd7,
                        0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(CHAP_Response54, 49)
```

/* HDLC Frame: Response (CHAP, ACFC, PFC, ACCM=0, Identifier = 1)

0x18, 0x01	SDU length (35 Byte = 280 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0xc2, 0x23	Protocol = CHAP
0x02	CHAP Code (Response)
0x01	Identifier
0x00, 0x1d	Length: 29
0x10	Value-Size: 16
0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3, 0x13, 0x08, 0xc2, 0x2f, 0xd7	Value
0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	Name "testname"
0xf0, 0xb8	FCS (dummy)
0x7e	Flag

*/

```
FIELD(CHAP_Response76)
        0x18, 0x01,
        0x00, 0x00,
        0x7e,
```

```
        0xc2, 0x23,
        0x02,
        0x01,
        0x00, 0x1d,
        0x10,
        0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3, 0x13, 0x08, 0xc2, 0x2f,
0xd7,
        0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65,
        0xf0, 0xb8,
        0x7e
ENDFIELD(CHAP_Response76, 39)

/*
```

3.3.12 CHAP Success

```
*/
/* HDLC Frame: Success  (CHAP, ACFC, PFC, Identifier = 1)
```

0x70, 0x00		SDU length (14 Byte = 112 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xc2, 0x23	0xc2, 0x23	Protocol = CHAP
0x03	0x7d, 0x23	CHAP Code (Success)
0x01	0x7d, 0x21	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length: 4
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

```
*/
FIELD(CHAP_Success54)
                        0x70, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xc2, 0x23,
                        0x7d, 0x23,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x24,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(CHAP_Success54, 18)

/* HDLC Frame: Success  (CHAP, ACFC, PFC, ACCM=0, Identifier = 1)
```

0x50, 0x00	SDU length (10 Byte = 80 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0xc2, 0x23	Protocol = CHAP
0x03	CHAP Code (Success)
0x01	Identifier
0x00, 0x04	Length: 4
0xf0, 0xb8	FCS (dummy)
0x7e	Flag

```
*/
FIELD(CHAP_Success76)
        0x50, 0x00,
        0x00, 0x00,
        0x7e,
        0xc2, 0x23,
        0x03,
```

```
        0x01,
        0x00, 0x04,
        0xf0, 0xb8,
        0x7e
ENDFIELD(CHAP_Success76, 14)

/*
```

3.3.13 PAP Authentication Request

```
*/
/* HDLC Frame: Authenticate Request  (PAP, Client Mode)
```

0x20, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x01	0x7d, 0x21	PAP Code (Auth Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x016	0x7d, 0x20, 0x7d, 0x36	Length
0x08	0x7d, 0x28	Peer-ID Length
0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	Peer-ID “testname”
0x08	0x7d, 0x28	Password Length
0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	Password “testpass”
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

```
*/
FIELD(AuthRequest0)  0x28, 0x01,
                     0x00, 0x00,
                     0x7e,
                     0xff,
                     0x7d, 0x23,
                     0xc0, 0x23,
                     0x7d, 0x21,
                     0x7d, 0x21,
                     0x7d, 0x20, 0x7d, 0x36,
                     0x7d, 0x28,
                     0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65,
                     0x7d, 0x28,
                     0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73,
                     0xf0, 0xb8,
                     0x7e
ENDFIELD(AuthRequest0, 41)

/* HDLC Frame: Authenticate Request  (PAP, Client Mode, ACFC, PFC)
```

0x10, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x01	0x7d, 0x21	PAP Code (Auth Request)

0x01	0x7d, 0x21	Identifier
0x00, 0x016	0x7d, 0x20, 0x7d, 0x36	Length
0x08	0x7d, 0x28	Peer-ID Length
0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	Peer-ID “testname”
0x08	0x7d, 0x28	Password Length
0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	Password “testpass”
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(AuthRequest0_C)        0x10, 0x01,
                             0x00, 0x00,
                             0x7e,
                             0xc0, 0x23,
                             0x7d, 0x21,
                             0x7d, 0x21,
                             0x7d, 0x20, 0x7d, 0x36,
                             0x7d, 0x28,
                             0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65,
                             0x7d, 0x28,
                             0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73,
                             0xf0, 0xb8,
                             0x7e
ENDFIELD(AuthRequest0_C, 38)
```

/* HDLC Frame: Authenticate Request (PAP, Client Mode, Identifier 2)

0x20, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x01	0x7d, 0x21	PAP Code (Auth Request)
0x02	0x7d, 0x22	Identifier
0x00, 0x016	0x7d, 0x20, 0x7d, 0x36	Length
0x08	0x7d, 0x28	Peer-ID Length
0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	Peer-ID “testname”
0x08	0x7d, 0x28	Password Length
0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	Password “testpass”
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(AuthRequest1)   0x28, 0x01,
                      0x00, 0x00,
```

```
                0x7e,
                0xff,
                0x7d, 0x23,
                0xc0, 0x23,
                0x7d, 0x21,
                0x7d, 0x22,
                0x7d, 0x20, 0x7d, 0x36,
                0x7d, 0x28,
                0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65,
                0x7d, 0x28,
                0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73,
                0xf0, 0xb8,
                0x7e
ENDFIELD(AuthRequest1, 41)

/* HDLC Frame: Authenticate Request  (PAP, ACFC, PFC, Identifier = 0)
```

0x10, 0x01		SDU length (34 Byte = 272 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x01	0x7d, 0x21	PAP Code (Authenticate Request)
0x00	0x7d, 0x20	Identifier
0x00, 0x016	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x08	0x7d, 0x28	Peer-ID Length: 8
0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65	Peer-ID “testname”
0x08	0x7d, 0x28	Password Length: 8
0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	Password “testpass”
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(PAP_AuthRequest67)
        0x10, 0x01,
        0x00, 0x00,
        0x7e,
        0xc0, 0x23,
        0x7d, 0x21,
        0x7d, 0x20,
        0x7d, 0x20, 0x7d, 0x36,
        0x7d, 0x28,
        0x74, 0x65, 0x73, 0x74, 0x6E, 0x61, 0x6D, 0x65,
        0x7d, 0x28,
        0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73,
        0xf0, 0xb8,
        0x7e
ENDFIELD(PAP_AuthRequest67, 38)

/*
```

3.3.14 PAP Authentication NAK

```
*/
```

/* HDLC Frame: Authenticate Nack (PAP, Client Mode, Identifier 1)

0x98, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x03	0x7d, 0x23	PAP Code (Auth Nack)
0x01	0x7d, 0x21	Identifier
0x00, 0x05	0x7d, 0x20, 0x7d, 0x25	Length
0x00	0x7d, 0x20	Message Length
		Message (None)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(AuthNack0)        0x98, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0xc0, 0x23,
                        0x7d, 0x23,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x25,
                        0x7d, 0x20,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(AuthNack0, 23)

/*
```

3.3.15 PAP Authentication Ack

*/

/* HDLC Frame: Authenticate Ack (PAP, Client Mode)

0x98, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x02	0x7d, 0x22	PAP Code (Auth Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x05	0x7d, 0x20, 0x7d, 0x25	Length
0x00	0x7d, 0x20	Message Length
		Message (None)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(AuthAck0)             0x98, 0x00,
                        0, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0xc0, 0x23,
```

```
                    0x7d, 0x22,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x25,
                    0x7d, 0x20,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(AuthAck0, 23)
```

/* HDLC Frame: Authenticate Ack (PAP, Client Mode, ACFC, PFC)

0x80, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x02	0x7d, 0x22	PAP Code (Auth Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x05	0x7d, 0x20, 0x7d, 0x25	Length
0x00	0x7d, 0x20	Message Length
		Message (None)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(AuthAck0_C)       0x80, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xc0, 0x23,
                    0x7d, 0x22,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x25,
                    0x7d, 0x20,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(AuthAck0_C, 20)
```

/* HDLC Frame: Authenticate Ack (PAP, ACFC, PFC, Identifier = 0)

0x80, 0x00		SDU length (16 Byte = 128 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xc0, 0x23	0xc0, 0x23	Protocol = PAP
0x02	0x7d, 0x22	PAP Code (Authenticate Ack)
0x00	0x7d, 0x20	Identifier
0x00, 0x05	0x7d, 0x20, 0x7d, 0x25	Length: 5
0x00	0x7d, 0x20	Message Length
		Message (None)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(PAP_AuthAck67)
        0x80, 0x00,
        0x00, 0x00,
        0x7e,
        0xc0, 0x23,
        0x7d, 0x22,
        0x7d, 0x20,
        0x7d, 0x20, 0x7d, 0x25,
        0x7d, 0x20,
```

```
        0xf0, 0xb8,
        0x7e
ENDFIELD(PAP_AuthAck67, 20)

/*
```

3.3.16 CCP Configure Request

```
*/
/* HDLC Frame: CCP Configure Request (Identifier = 1)
```

0xe8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0xfd	0x80, 0xfd	Protocol = CCP
0x01	0x7d, 0x21	CCP Code (Conf Req)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (OUI)
0x06	0x7d, 0x26	Option Length
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (dummy)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(CCPConfReq0)  0xe8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xff,
                    0x7d, 0x23,
                    0x80, 0xfd,
                    0x7d, 0x21,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x2a,
                    0x7d, 0x23,
                    0x7d, 0x26,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(CCPConfReq0, 33)

/* HDLC Frame: Configure Request (CCP, ACFC, PFC, Identifier = 4)
```

0x28, 0x0c		SDU length (389 Byte = 3112 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0xfd	0x80, 0xfd	Protocol = CCP
0x01	0x7d, 0x21	CCP Code (Configure Request)
0x04	0x7d, 0x24	Identifier
0x00, 0xd5	0x7d, 0x20, 0xd5	Length: 213
…	…	Content
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(CCP_ConfRequest59)
                    0x28, 0x0c,
                    0x00, 0x00,
                    0x7e,
```

```
                    0x80, 0xfd,
                    0x7d, 0x21,
                    0x7d, 0x24,
                    0x7d, 0x20, 0xd5,
                    0xfe, 0xcb, 0x7d, 0x21, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x74, 0xff,
                    0xc9, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x68,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0xa0, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x30, 0x30, 0x20, 0x62, 0x7d, 0x25, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0xc0, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0xa4, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0xbc, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x38,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x70, 0x7d, 0x20, 0xea, 0x7d, 0x20,
                    0xb0, 0xc7, 0x7d, 0x34, 0x7d, 0x20, 0x69, 0xcc, 0xf1, 0x77, 0x7d, 0x20,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0xe8, 0xfe, 0xc9, 0x7d, 0x20, 0xc8, 0xcd, 0xf1, 0x77, 0x7d,
                    0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d,
                    0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x74, 0x7d, 0x20, 0x6e, 0x7d,
                    0x20, 0x7d, 0x3d, 0x7d, 0x20, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21,
                    0x7d, 0x2b, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0xdb, 0x80, 0xfd, 0x7d, 0x21, 0x7d, 0x24, 0x7d, 0x20, 0xd5,
                    0xfe, 0xcb, 0x7d, 0x21, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x74, 0xff,
                    0xc9, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x68,
                    0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x32, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20,
                    0x7d, 0x20, 0x7d, 0x26,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(CCP_ConfRequest59, 393)

/* HDLC Frame: Configure Request (CCP, ACFC, PFC, ACCM=0, Identifier = 4)
```

0xd8, 0x06	SDU length (219 Byte = 1752 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0xfd	Protocol = CCP
0x01	CCP Code (Configure Request)
0x04	Identifier
0x00, 0xd5	Length: 213
…	Content
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
*/
FIELD(CCP_ConfRequest81)
        0xd8, 0x06,
        0x00, 0x00,
        0x7e,
        0x80, 0xfd,
        0x01,
```

```
        0x04,
        0x00, 0xd5,
        0xfe, 0xcb, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x00, 0x00, 0x74, 0xff, 0xc9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68, 0x00, 0x00, 0x00, 0xa0,
0x00,
        0x00, 0x00, 0x30, 0x30, 0x20, 0x62, 0x05, 0x00, 0x00, 0x00, 0xc0, 0x00, 0x00, 0x00, 0xa4,
0x00,
        0x00, 0x00, 0xbc, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x00, 0x70, 0x00, 0xea, 0x00, 0xb0,
0xc7,
        0x14, 0x00, 0x69, 0xcc, 0xf1, 0x77, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0xe8,
0xfe,
        0xc9, 0x00, 0xc8, 0xcd, 0xf1, 0x77, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x74, 0x00, 0x6e, 0x00, 0x1d, 0x00, 0x20, 0x00, 0x00,
0x01,
        0x0b, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xdb, 0x80, 0xfd, 0x01, 0x04, 0x00, 0xd5, 0xfe,
0xcb,
        0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00,
        0x74, 0xff, 0xc9, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68, 0x00, 0x00, 0x12, 0x06, 0x00, 0x00,
0x00,
        0x06,
        0xf0, 0xb8,
        0x7e
ENDFIELD(CCP_ConfRequest81, 223)

/*
```

3.3.17 IPCP Configure Request

```
*/
/* HDLC Frame: IPCP Configure Request (Identifier = 1)
```

0xe8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (IP-Adress 0.0.0.0)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPCPReq0)         0xe8, 0x00,
                        0x00, 0x00,
                        0x7e,
```

```
                0xff,
                0x7d, 0x23,
                0x80, 0x21,
                0x7d, 0x21,
                0x7d, 0x21,
                0x7d, 0x20, 0x7d, 0x2a,
                0x7d, 0x23,
                0x7d, 0x26,
                0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                0xf0, 0xb8,
                0x7e
ENDFIELD(IPCPReq0, 33)

/* HDLC Frame: IPCP Configure Request (ACFC, PFC, Identifier = 1)
```

0xd0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Option Value (IP-Adress 0.0.0.0)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPCPReq0_C)
        0xd0, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x21,
        0x7d, 0x21,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x23,
        0x7d, 0x26,
        0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCPReq0_C, 30)

/* HDLC Frame: IPCP Configure Req (Identifier = 2)
```

0xd8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x02	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0xc6, 0xa8, 0x01, 0x05	0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25	Option Value (IP-Adress 192.168.1.5)

0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCPReq1)             0xd8, 0x00,
                            0x00, 0x00,
                            0x7e,
                            0xff,
                            0x7d, 0x23,
                            0x80, 0x21,
                            0x7d, 0x21,
                            0x7d, 0x22,
                            0x7d, 0x20, 0x7d, 0x2a,
                            0x7d, 0x23,
                            0x7d, 0x26,
                            0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25,
                            0xf0, 0xb8,
                            0x7e
ENDFIELD(IPCPReq1, 31)
```

/* HDLC Frame: IPCP Configure Req (ACFC, PFC, Identifier = 2)

0xc0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x02	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0xc6, 0xa8, 0x01, 0x05	0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25	Option Value (IP-Adress 192.168.1.5)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCPReq1_C)           0xc0, 0x00,
                            0x00, 0x00,
                            0x7e,
                            0x80, 0x21,
                            0x7d, 0x21,
                            0x7d, 0x22,
                            0x7d, 0x20, 0x7d, 0x2a,
                            0x7d, 0x23,
                            0x7d, 0x26,
                            0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25,
                            0xf0, 0xb8,
                            0x7e
ENDFIELD(IPCPReq1_C, 28)
```

/* HDLC Frame: IPCP Configure Req (Identifier = 5)

0x88, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x05	0x7d, 0x25	Identifier

0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCPReq2)
0x88, 0x00,
0x00, 0x00,
0x7e,
0xff,
0x7d, 0x23,
0x80, 0x21,
0x7d, 0x21,
0x7d, 0x25,
0x7d, 0x20, 0x7d, 0x24,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCPReq2, 21)

/* HDLC Frame: IPCP Configure Req (ACFC, PFC, Identifier = 5)

0x70, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x05	0x7d, 0x25	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCPReq2_C)
0x70, 0x00,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x21,
0x7d, 0x25,
0x7d, 0x20, 0x7d, 0x24,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCPReq2_C, 18)

/* HDLC Frame: IPCP Configure Req (ACFC, PFC, IP address, Identifier = 5)

0xD0, 0x00		SDU length (26 Byte = 208 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Conf Req)
0x05	0x7d, 0x25	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x0a, 0x00, 0x00, 0x01	0x7d, 0x2a, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21	Static Address (10.0.0.1)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfRequest3_C)

```
        0xD0, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x21,
        0x7d, 0x25,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x2a, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfRequest3_C, 30)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, Identifier = 5)
```

0x80, 0x02		SDU length (80 Byte = 640 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x05	0x7d, 0x25	Identifier
0x00, 0x28	0x7d, 0x20, 0x28	Length: 40
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x21	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x82	0x82	Primary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x84	0x84	Secondary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfRequest61)
                        0x80, 0x02,
                        0x00, 0x00,
                        0x7e,
                        0x80, 0x21,
                        0x7d, 0x21,
                        0x7d, 0x25,
                        0x7d, 0x20, 0x28,
```

```
                    0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
                    0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x81, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x82, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x83, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x84, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCP_ConfRequest61, 84)
```

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, Identifier = 1)

0xc8, 0x00		SDU length (25 Byte = 200 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfRequest62)
                    0xc8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x80, 0x21,
                    0x7d, 0x21,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x2a,
                    0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCP_ConfRequest62, 29)
```

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, Identifier = 6)

0xd8, 0x01		SDU length (59 Byte = 472 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x06	0x7d, 0x26	Identifier
0x00, 0x1c	0x7d, 0x20, 0x7d, 0x3c	Length: 28
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00,	0x7d, 0x20, 0x7d, 0x20,	Dynamic Address

0x00, 0x00	0x7d, 0x20, 0x7d, 0x20	
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfRequest63)
                        0xd8, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0x80, 0x21,
                        0x7d, 0x21,
                        0x7d, 0x26,
                        0x7d, 0x20, 0x7d, 0x3c,
                        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
                        0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                        0x81, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                        0x83, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPCP_ConfRequest63, 63)
```

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, Identifier = 7)

0x90, 0x01		SDU length (50 Byte = 400 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x07	0x7d, 0x27	Identifier
0x00, 0x1c	0x7d, 0x20, 0x7d, 0x3c	Length: 28
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x18, 0x81	0x8d, 0x40, 0x7d, 0x38, 0x81	Address 141.64.24.129
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfRequest65)
                        0x90, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0x80, 0x21,
                        0x7d, 0x21,
                        0x7d, 0x27,
                        0x7d, 0x20, 0x7d, 0x3c,
                        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
                        0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
                        0x81, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x38, 0x81,
                        0x83, 0x7d, 0x26, 0x8d, 0x40, 0xfe, 0x83,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPCP_ConfRequest65, 54)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, VJ with uncompressed CSID, Identifier = 5)
```

0x80, 0x02		SDU length (80 Byte = 640 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x05	0x7d, 0x25	Identifier
0x00, 0x28	0x7d, 0x20, 0x28	Length: 40
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x00	0x7d, 0x20	Comp-Slot-ID: uncompressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x82	0x82	Primary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x84	0x84	Secondary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfRequest69)
                        0x80, 0x02,
                        0x00, 0x00,
```

```
                    0x7e,
                    0x80, 0x21,
                    0x7d, 0x21,
                    0x7d, 0x25,
                    0x7d, 0x20, 0x28,
                    0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x20,
                    0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x81, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x82, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x83, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0x84, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCP_ConfRequest69, 84)
```

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without Header Compression, Identifier = 1)

0x70, 0x00		SDU length (14 Byte = 112 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x01	0x7d, 0x21	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length: 4
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfRequest70)
                    0x70, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x80, 0x21,
                    0x7d, 0x21,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x24,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCP_ConfRequest70, 18)
```

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without Header Compression, Identifier = 6)

0x80, 0x01		SDU length (48 Byte = 384 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x06	0x7d, 0x26	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00,	0x7d, 0x20, 0x7d, 0x20,	Dynamic Address

0x00, 0x00	0x7d, 0x20, 0x7d, 0x20	
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfRequest71)
0x80, 0x01,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x21,
0x7d, 0x26,
0x7d, 0x20, 0x7d, 0x36,
0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x81, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x83, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfRequest71, 52)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without Header Compression, Identifier = 7)

0x38, 0x01		SDU length (39 Byte = 312 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x07	0x7d, 0x27	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x18, 0x81	0x8d, 0x40, 0x7d, 0x38, 0x81	Address 141.64.24.129
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfRequest73)
0x38, 0x01,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x21,
0x7d, 0x27,
0x7d, 0x20, 0x7d, 0x36,
0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
0x81, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x38, 0x81,
0x83, 0x7d, 0x26, 0x8d, 0x40, 0xfe, 0x83,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfRequest73, 43)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, ACCM=0, Identifier = 5)

0x70, 0x01	SDU length (46 Byte = 368 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x01	IPCP Code (Configure Request)
0x05	Identifier
0x00, 0x28	Length: 40
0x02	IP-Header Compression
0x06	Option Length: 6
0x00, 0x2d	Van Jacobson Header Compression
0x0f	Max-Slot-ID: 15
0x01	Comp-Slot-ID: compressed slot ID
0x03	IP-Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x81	Primary DNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x82	Primary NBNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x83	Secondary DNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x84	Secondary NBNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
 */
FIELD(IPCP_ConfRequest83)
        0x70, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x01,
        0x05,
        0x00, 0x28,
        0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01,
        0x03, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x81, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x82, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x83, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x84, 0x06, 0x00, 0x00, 0x00, 0x00,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfRequest83, 50)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, ACCM=0, Identifier = 1)
```

0x80, 0x00	SDU length (16 Byte = 128 Bit)
0x00, 0x00	SDU offset

0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x01	IPCP Code (Configure Request)
0x01	Identifier
0x00, 0x0a	Length: 10
0x02	IP-Header Compression
0x06	Option Length: 6
0x00, 0x2d	Van Jacobson Header Compression
0x0f	Max-Slot-ID: 15
0x01	Comp-Slot-ID: compressed slot ID
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

*/
FIELD(IPCP_ConfRequest84)
0x80, 0x00,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x01,
0x01,
0x00, 0x0a,
0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfRequest84, 20)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, ACCM=0, Identifier = 6)

0x10, 0x01	SDU length (34 Byte = 272 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x01	IPCP Code (Configure Request)
0x06	Identifier
0x00, 0x1c	Length: 28
0x02	IP-Header Compression
0x06	Option Length: 6
0x00, 0x2d	Van Jacobson Header Compression
0x0f	Max-Slot-ID: 15
0x01	Comp-Slot-ID: compressed slot ID
0x03	IP-Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x81	Primary DNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x83	Secondary DNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

*/
FIELD(IPCP_ConfRequest85)

```
        0x10, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x01,
        0x06,
        0x00, 0x1c,
        0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01,
        0x03, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x81, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x83, 0x06, 0x00, 0x00, 0x00, 0x00,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfRequest85, 38)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, ACCM=0, Identifier = 7)
```

0x10, 0x01	SDU length (34 Byte = 272 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x01	IPCP Code (Configure Request)
0x07	Identifier
0x00, 0x1c	Length: 28
0x02	IP-Header Compression
0x06	Option Length: 6
0x00, 0x2d	Van Jacobson Header Compression
0x0f	Max-Slot-ID: 15
0x01	Comp-Slot-ID: compressed slot ID
0x03	IP-Address
0x06	Option Length: 6
0x8d, 0x40, 0x15, 0x02	Address 141.64.21.2
0x81	Primary DNS Server Address
0x06	Option Length: 6
0x8d, 0x40, 0x18, 0x81	Address 141.64.24.129
0x83	Secondary DNS Server Address
0x06	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
 */
FIELD(IPCP_ConfRequest87)
        0x10, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x01,
        0x07,
        0x00, 0x1c,
        0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01,
        0x03, 0x06, 0x8d, 0x40, 0x15, 0x02,
        0x81, 0x06, 0x8d, 0x40, 0x18, 0x81,
        0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83,
        0xf0, 0xb8,
        0x7e
```

ENDFIELD(IPCP_ConfRequest87, 38)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without Header Compression, Identifier = 3)

0x70, 0x00		SDU length (14 Byte = 112 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x03	0x7d, 0x23	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length: 4
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfRequest88)
0x70, 0x00,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x21,
0x7d, 0x23,
0x7d, 0x20, 0x7d, 0x24,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfRequest88, 18)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without DNS, Identifier = 6)

0x28, 0x01		SDU length (37 Byte = 296 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x06	0x7d, 0x26	Identifier
0x00, 0x10	0x7d, 0x20, 0x7d, 0x30	Length: 16
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfRequest89)
0x28, 0x01,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x21,
0x7d, 0x26,
0x7d, 0x20, 0x7d, 0x30,
0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0xf0, 0xb8,

```
        0x7e
ENDFIELD(IPCP_ConfRequest89, 41)

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without DNS, Identifier = 7)
```

0x18, 0x01		SDU length (35 Byte = 280 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x01	0x7d, 0x21	IPCP Code (Configure Request)
0x07	0x7d, 0x27	Identifier
0x00, 0x10	0x7d, 0x20, 0x7d, 0x30	Length: 16
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfRequest91)
        0x18, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x21,
        0x7d, 0x27,
        0x7d, 0x20, 0x7d, 0x30,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
        0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfRequest91, 39)

/*
```

3.3.18 IPCP Configure Reject

```
*/
/* HDLC Frame: Configure Reject (IPCP, ACFC, PFC, Identifier = 5)
```

0x20, 0x01		SDU length (36 Byte = 288 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x04	0x7d, 0x24	IPCP Code (Configure Reject)
0x05	0x7d, 0x25	Identifier
0x00, 0x10	0x7d, 0x20, 0x7d, 0x30	Length: 16
0x82	0x82	Primary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x84	0x84	Secondary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6

0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfReject61)
0x20, 0x01,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x24,
0x7d, 0x25,
0x7d, 0x20, 0x7d, 0x30,
0x82, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x84, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfReject61, 40)

/* HDLC Frame: Configure Reject (IPCP, ACFC, PFC, include Header Compression, Identifier = 5)

0x78, 0x01		SDU length (47 Byte = 376 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x04	0x7d, 0x24	IPCP Code (Configure Reject)
0x05	0x7d, 0x25	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x02	0x7d, 0x22	Header Compression
0x06	0c7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x21	Comp-Slot-ID: compressed slot ID
0x82	0x82	Primary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x84	0x84	Secondary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPCP_ConfReject61F)
0x78, 0x01,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x24,
0x7d, 0x25,
0x7d, 0x20, 0x7d, 0x36,
0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
0x82, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0x84, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
0xf0, 0xb8,
0x7e

ENDFIELD(IPCP_ConfReject61F, 51)

/* HDLC Frame: Configure Reject (IPCP, ACFC, PFC, reject DNS, Identifier = 5)

0xd0, 0x01		SDU length (58 Byte = 464 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x04	0x7d, 0x24	IPCP Code (Configure Reject)
0x05	0x7d, 0x25	Identifier
0x00, 0x1c	0x7d, 0x20, 0x7d, 0x3c	Length: 28
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x82	0x82	Primary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x84	0x84	Secondary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfReject61DNS)
        0xd0, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x24,
        0x7d, 0x25,
        0x7d, 0x20, 0x7d, 0x3c,
        0x81, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x82, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x83, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x84, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfReject61DNS, 62)
```

/* HDLC Frame: Configure Reject (IPCP, ACFC, PFC, include Header Compression, Identifier = 5)

0x78, 0x01		SDU length (47 Byte = 376 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x04	0x7d, 0x24	IPCP Code (Configure Reject)
0x05	0x7d, 0x25	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x02	0x7d, 0x22	Header Compression
0x06	0c7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression

0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x00	0x7d, 0x20	Comp-Slot-ID: uncompressed slot ID
0x82	0x82	Primary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x84	0x84	Secondary NBNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfReject69)
        0x78, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x24,
        0x7d, 0x25,
        0x7d, 0x20, 0x7d, 0x36,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x20,
        0x82, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x84, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfReject69, 51)
```

/* HDLC Frame: Configure Reject (IPCP, ACFC, PFC, without Header Compression, Identifier = 6)

0x80, 0x01		SDU length (48 Byte = 384 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x04	0x7d, 0x24	IPCP Code (Configure Reject)
0x06	0x7d, 0x26	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x00, 0x00, 0x00	0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20	Dynamic Address
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfReject71)
        0x80, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
```

```
        0x7d, 0x24,
        0x7d, 0x26,
        0x7d, 0x20, 0x7d, 0x36,
        0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x81, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0x83, 0x7d, 0x26, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x20,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfReject71, 52)

/* HDLC Frame: Configure Reject (IPCP, ACFC, PFC, ACCM=0, Identifier = 5)
```

0xb0, 0x00	SDU length (22 Byte = 176 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x04	IPCP Code (Configure Reject)
0x05	Identifier
0x00, 0x10	Length: 16
0x82	Primary NBNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0x84	Secondary NBNS Server Address
0x06	Option Length: 6
0x00, 0x00, 0x00, 0x00	Dynamic Address
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
 */
FIELD(IPCP_ConfReject83)
        0xb0, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x04,
        0x05,
        0x00, 0x10,
        0x82, 0x06, 0x00, 0x00, 0x00, 0x00,
        0x84, 0x06, 0x00, 0x00, 0x00, 0x00,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfReject83, 26)

/*
```

3.3.19 IPCP Configure NAK

```
*/
/* HDLC Frame: IPCP Configure Nack (Identifier = 1)
```

0xd8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x03	0x7d, 0x23	IPCP Code (Conf Nack)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length

0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0xc6, 0xa8, 0x01, 0x05	0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25	Option Value (IP-Adress 192.168.1.5)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCPNack0)    0xd8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0xff,
                    0x7d, 0x23,
                    0x80, 0x21,
                    0x7d, 0x23,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x2a,
                    0x7d, 0x23,
                    0x7d, 0x26,
                    0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCPNack0, 31)
```

/* HDLC Frame: IPCP Configure Nack (ACFC, PFC, Identifier = 1)

0xc0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x03	0x7d, 0x23	IPCP Code (Conf Nack)
0x01	0x7d, 0x21	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0xc6, 0xa8, 0x01, 0x05	0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25	Option Value (IP-Adress 192.168.1.5)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCPNack0_C)  0xc0, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x80, 0x21,
                    0x7d, 0x23,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x2a,
                    0x7d, 0x23,
                    0x7d, 0x26,
                    0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCPNack0_C, 28)
```

/* HDLC Frame: Configure NAK (IPCP, ACFC, PFC, Identifier = 6)

0x38, 0x01		SDU length (39 Byte = 312 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP

0x03	0x7d, 0x23	IPCP Code (Configure NAK)
0x06	0x7d, 0x26	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x18, 0x81	0x8d, 0x40, 0x7d, 0x38, 0x81	Address 141.64.24.129
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfNAK63)
                        0x38, 0x01,
                        0x00, 0x00,
                        0x7e,
                        0x80, 0x21,
                        0x7d, 0x23,
                        0x7d, 0x26,
                        0x7d, 0x20, 0x7d, 0x36,
                        0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
                        0x81, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x38, 0x81,
                        0x83, 0x7d, 0x26, 0x8d, 0x40, 0xfe, 0x83,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPCP_ConfNAK63, 43)
```

/* HDLC Frame: Configure NAK (IPCP, ACFC, PFC, Identifier = 6)

0x38, 0x01		SDU length (39 Byte = 312 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x03	0x7d, 0x23	IPCP Code (Configure NAK)
0x06	0x7d, 0x26	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x18, 0x81	0x8d, 0x40, 0x7d, 0x38, 0x81	Address 141.64.24.129
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPCP_ConfNAK71)
```

```
                0x38, 0x01,
                0x00, 0x00,
                0x7e,
                0x80, 0x21,
                0x7d, 0x23,
                0x7d, 0x26,
                0x7d, 0x20, 0x7d, 0x36,
                0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
                0x81, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x38, 0x81,
                0x83, 0x7d, 0x26, 0x8d, 0x40, 0xfe, 0x83,
                0xf0, 0xb8,
                0x7e
ENDFIELD(IPCP_ConfNAK71, 43)
```

/* HDLC Frame: Configure NAK (IPCP, ACFC, PFC, ACCM=0, Identifier = 6)

0xe0, 0x00	SDU length (28 Byte = 224 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x03	IPCP Code (Configure NAK)
0x06	Identifier
0x00, 0x16	Length: 22
0x03	IP-Address
0x06	Option Length: 6
0x8d, 0x40, 0x15, 0x02	Address 141.64.21.2
0x81	Primary DNS Server Address
0x06	Option Length: 6
0x8d, 0x40, 0x18, 0x81	Address 141.64.24.129
0x83	Secondary DNS Server Address
0x06	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
 */
FIELD(IPCP_ConfNAK85)
        0xe0, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x03,
        0x06,
        0x00, 0x16,
        0x03, 0x06, 0x8d, 0x40, 0x15, 0x02,
        0x81, 0x06, 0x8d, 0x40, 0x18, 0x81,
        0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfNAK85, 32)
```

/* HDLC Frame: Configure NAK (IPCP, ACFC, PFC, without DNS, Identifier = 6)

0xc0, 0x00		SDU length (24 Byte = 192 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP

0x03	0x7d, 0x23	IPCP Code (Configure NAK)
0x06	0x7d, 0x26	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfNAK89)
        0xc0, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x23,
        0x7d, 0x26,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfNAK89, 28)

/*
```

3.3.20 IPCP Configure Ack

```
*/
/* HDLC Frame: IPCP Configure Ack (Identifier = 2)
```

0xd8, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Conf Ack)
0x01	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0xc6, 0xa8, 0x01, 0x05	0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25	Option Value (IP-Adress 192.168.1.5)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPCPAck1)         0xd8, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0x80, 0x21,
                        0x7d, 0x22,
                        0x7d, 0x22,
                        0x7d, 0x20, 0x7d, 0x2a,
                        0x7d, 0x23,
                        0x7d, 0x26,
                        0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25,
                        0xf0, 0xb8,
```

```
                    0x7e
ENDFIELD(IPCPAck1, 31)
```

/* HDLC Frame: IPCP Configure Ack (ACFC, PFC, Identifier = 2)

0xc0, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Conf Ack)
0x01	0x7d, 0x22	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length
0x03	0x7d, 0x23	Option Type (IP-Adress)
0x06	0x7d, 0x26	Option Length
0xc6, 0xa8, 0x01, 0x05	0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25	Option Value (IP-Adress 192.168.1.5)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPCPAck1_C)           0xc0, 0x00,
                            0x00, 0x00,
                            0x7e,
                            0x80, 0x21,
                            0x7d, 0x22,
                            0x7d, 0x22,
                            0x7d, 0x20, 0x7d, 0x2a,
                            0x7d, 0x23,
                            0x7d, 0x26,
                            0xc6, 0xa8, 0x7d, 0x21, 0x7d, 0x25,
                            0xf0, 0xb8,
                            0x7e
ENDFIELD(IPCPAck1_C, 28)
```

/* HDLC Frame: IPCP Configure Ack (Identifier = 5)

0x88, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Conf Req)
0x05	0x7d, 0x25	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCPAck2)             0x88, 0x00,
                            0x00, 0x00,
                            0x7e,
                            0xff,
                            0x7d, 0x23,
                            0x80, 0x21,
                            0x7d, 0x22,
                            0x7d, 0x25,
                            0x7d, 0x20, 0x7d, 0x24,
                            0xf0, 0xb8,
                            0x7e
ENDFIELD(IPCPAck2, 21)
```

/* HDLC Frame: IPCP Configure Ack (ACFC, PFC, Identifier = 5)

0x70, 0x00		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Conf Ack)
0x05	0x7d, 0x25	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCPAck2_C)
        0x70, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x22,
        0x7d, 0x25,
        0x7d, 0x20, 0x7d, 0x24,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCPAck2_C, 18)
```

/* HDLC Frame: IPCP Configure Ack (ACFC, PFC, IP address, Identifier = 5)

0xD0, 0x00		SDU length (26 Byte = 208 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Conf Ack)
0x05	0x7d, 0x25	Identifier
0x00, 0x0a	0x7d, 0x20, 0x7d, 0x2a	Length: 10
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x0a, 0x00, 0x00, 0x01	0x7d, 0x2a, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21	Static Address (10.0.0.1)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfAck3_C)
        0xD0, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x22,
        0x7d, 0x25,
        0x7d, 0x20, 0x7d, 0x2a,
        0x7d, 0x23, 0x7d, 0x26, 0x7d, 0x2a, 0x7d, 0x20, 0x7d, 0x20, 0x7d, 0x21,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfAck3_C, 30)
```

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, Identifier = 1)

0xc8, 0x00		SDU length (25 Byte = 200 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP

0x02	0x7d, 0x22	IPCP Code (Configure Ack)
0x05	0x7d, 0x25	Identifier
0x00, 0x0a	0x7d, 0x20, 0x2a	Length: 10
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPCP_ConfAck62)
                        0xc8, 0x00,
                        0x00, 0x00,
                        0x7e,
                        0x80, 0x21,
                        0x7d, 0x22,
                        0x7d, 0x21,
                        0x7d, 0x20, 0x7d, 0x2a,
                        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPCP_ConfAck62, 29)

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, Identifier = 7)
```

0x90, 0x01		SDU length (50 Byte = 400 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Configure Ack)
0x07	0x7d, 0x27	Identifier
0x00, 0x1c	0x7d, 0x20, 0x7d, 0x3c	Length: 28
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x18, 0x81	0x8d, 0x40, 0x7d, 0x38, 0x81	Address 141.64.24.129
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPCP_ConfAck65)
                        0x90, 0x01,
                        0x00, 0x00,
```

```
                    0x7e,
                    0x80, 0x21,
                    0x7d, 0x22,
                    0x7d, 0x27,
                    0x7d, 0x20, 0x7d, 0x3c,
                    0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
                    0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
                    0x81, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x38, 0x81,
                    0x83, 0x7d, 0x26, 0x8d, 0x40, 0xfe, 0x83,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCP_ConfAck65, 54)

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, without Header Compression, Identifier = 1)
```

0x70, 0x00		SDU length (14 Byte = 112 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Configure Ack)
0x01	0x7d, 0x21	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length: 4
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfAck70)
                    0xc8, 0x00,
                    0x00, 0x00,
                    0x7e,
                    0x80, 0x21,
                    0x7d, 0x22,
                    0x7d, 0x21,
                    0x7d, 0x20, 0x7d, 0x24,
                    0xf0, 0xb8,
                    0x7e
ENDFIELD(IPCP_ConfAck70, 18)

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, without Header Compression, Identifier = 7)
```

0x38, 0x01		SDU length (39 Byte = 312 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Configure Ack)
0x07	0x7d, 0x27	Identifier
0x00, 0x16	0x7d, 0x20, 0x7d, 0x36	Length: 22
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0x81	0x81	Primary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x18, 0x81	0x8d, 0x40, 0x7d, 0x38, 0x81	Address 141.64.24.129
0x83	0x83	Secondary DNS Server Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)

0x7e	Flag

*/
FIELD(IPCP_ConfAck73)
0x38, 0x01,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x7d, 0x22,
0x7d, 0x27,
0x7d, 0x20, 0x7d, 0x36,
0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
0x81, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x38, 0x81,
0x83, 0x7d, 0x26, 0x8d, 0x40, 0xfe, 0x83,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfAck73, 43)

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, ACCM=0, Identifier = 1)

0x80, 0x00	SDU length (16 Byte = 128 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x02	IPCP Code (Configure Ack)
0x01	Identifier
0x00, 0x0a	Length: 10
0x02	IP-Header Compression
0x06	Option Length: 6
0x00, 0x2d	Van Jacobson Header Compression
0x0f	Max-Slot-ID: 15
0x01	Comp-Slot-ID: compressed slot ID
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

*/
FIELD(IPCP_ConfAck84)
0x80, 0x00,
0x00, 0x00,
0x7e,
0x80, 0x21,
0x02,
0x01,
0x00, 0x0a,
0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01,
0xf0, 0xb8,
0x7e
ENDFIELD(IPCP_ConfAck84, 20)

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, ACCM=0, Identifier = 7)

0x10, 0x01	SDU length (34 Byte = 272 Bit)
0x00, 0x00	SDU offset
0x7e	Flag
0x80, 0x21	Protocol = IPCP
0x02	IPCP Code (Configure Ack)
0x07	Identifier
0x00, 0x1c	Length: 28
0x02	IP-Header Compression
0x06	Option Length: 6

0x00, 0x2d	Van Jacobson Header Compression
0x0f	Max-Slot-ID: 15
0x01	Comp-Slot-ID: compressed slot ID
0x03	IP-Address
0x06	Option Length: 6
0x8d, 0x40, 0x15, 0x02	Address 141.64.21.2
0x81	Primary DNS Server Address
0x06	Option Length: 6
0x8d, 0x40, 0x18, 0x81	Address 141.64.24.129
0x83	Secondary DNS Server Address
0x06	Option Length: 6
0x8d, 0x40, 0xfe, 0x83	Address 141.64.254.131
0xf0, 0xb8	FSC (dummy)
0x7e	Flag

```
 */
FIELD(IPCP_ConfAck87)
        0x10, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x02,
        0x07,
        0x00, 0x1c,
        0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01,
        0x03, 0x06, 0x8d, 0x40, 0x15, 0x02,
        0x81, 0x06, 0x8d, 0x40, 0x18, 0x81,
        0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfAck87, 38)
```

/* HDLC Frame: Configure Ack (IPCP, ACFC, PFC, without Header Compression, Identifier = 3)

0x70, 0x00		SDU length (14 Byte = 112 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Configure Ack)
0x03	0x7d, 0x23	Identifier
0x00, 0x04	0x7d, 0x20, 0x7d, 0x24	Length: 4
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
 */
FIELD(IPCP_ConfAck88)
        0x70, 0x00,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x22,
        0x7d, 0x23,
        0x7d, 0x20, 0x7d, 0x24,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfAck88, 18)
```

/* HDLC Frame: Configure Request (IPCP, ACFC, PFC, without DNS, Identifier = 7)

0x18, 0x01		SDU length (35 Byte = 280 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x80, 0x21	0x80, 0x21	Protocol = IPCP
0x02	0x7d, 0x22	IPCP Code (Configure Ack)
0x07	0x7d, 0x27	Identifier
0x00, 0x10	0x7d, 0x20, 0x7d, 0x30	Length: 16
0x02	0x7d, 0x22	IP-Header Compression
0x06	0x7d, 0x26	Option Length: 6
0x00, 0x2d	0x7d, 0x20, 0x2d	Van Jacobson Header Compression
0x0f	0x7d, 0x2f	Max-Slot-ID: 15
0x01	0x7d, 0x01	Comp-Slot-ID: compressed slot ID
0x03	0x7d, 0x23	IP-Address
0x06	0x7d, 0x26	Option Length: 6
0x8d, 0x40, 0x15, 0x02	0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22	Address 141.64.21.2
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
```
FIELD(IPCP_ConfAck91)
        0x18, 0x01,
        0x00, 0x00,
        0x7e,
        0x80, 0x21,
        0x7d, 0x22,
        0x7d, 0x27,
        0x7d, 0x20, 0x7d, 0x30,
        0x7d, 0x22, 0x7d, 0x26, 0x7d, 0x20, 0x2d, 0x7d, 0x2f, 0x7d, 0x21,
        0x7d, 0x23, 0x7d, 0x26, 0x8d, 0x40, 0x7d, 0x35, 0x7d, 0x22,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPCP_ConfAck91, 39)

/*
```

3.3.21 IP packets

*/
/* IP Paket: dummy packet A0

0x70, 0x03	SDU length
0x00, 0x00	SDU offset
0xa1, 0xa2, 0xa3, …., 0xaf	data (110 byte)

*/
```
FIELD(IPPACKET_A0)  0x70, 0x03,
                    0x00, 0x00,
                    0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                    0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
```

```
                        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
ENDFIELD(IPPACKET_A0,114)

/* IP Paket: dummy packet B0
```

0x70, 0x03	SDU length
0x00, 0x00	SDU offset
0xb1, 0xb2, 0xb3, …., 0xbf	data (110 byte)

```
*/
FIELD(IPPACKET_B0)  0x70, 0x03,
                        0x00, 0x00,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
ENDFIELD(IPPACKET_B0,114)

/* IP Paket: dummy packet C0
```

0x70, 0x03	SDU length
0x00, 0x00	SDU offset
0xc1, 0xc2, 0xc3, …., 0xcf	Data (110 byte)

```
*/
FIELD(IPPACKET_C0)  0x70, 0x03,
                        0x00, 0x00,
                        0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xb1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                        0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xcf,
ENDFIELD(IPPACKET_C0,114)

/* uncompressed TCP Paket: dummy packet D0
```

0xa0, 0x04	SDU length (148 Byte = 1184 Bit)
0x00, 0x00	SDU offset
….	Data (148 byte)

```
*/
FIELD(UTCPPACKET_D0)
        0xa0, 0x04,
        0x00, 0x00,
```

```
        0x45, 0x00, 0x00, 0x94, 0x04, 0x1e, 0x40, 0x00, 0x75, 0x00, 0xd9, 0x26, 0x3e, 0x60, 0xcf,
0x09,
        0x97, 0xbd, 0x82, 0xf8, 0x00, 0x50, 0x04, 0x10, 0xe2, 0x75, 0xdd, 0x43, 0x00, 0x1a, 0xbe,
0x78,
        0x50, 0x18, 0x21, 0x09, 0x18, 0xaf, 0x00, 0x00, 0x48, 0x54, 0x54, 0x50, 0x2f, 0x31, 0x2e,
0x31,
        0x20, 0x32, 0x30, 0x30, 0x20, 0x4f, 0x4b, 0x0d, 0x0a, 0x53, 0x65, 0x72, 0x76, 0x65, 0x72,
0x3a,
        0x20, 0x4d, 0x69, 0x63, 0x72, 0x6f, 0x73, 0x6f, 0x66, 0x74, 0x2d, 0x49, 0x49, 0x53, 0x2f,
0x34,
        0x2e, 0x30, 0x0d, 0x0a, 0x44, 0x61, 0x74, 0x65, 0x3a, 0x20, 0x46, 0x72, 0x69, 0x2c, 0x20,
0x30,
        0x36, 0x20, 0x4f, 0x63, 0x74, 0x20, 0x32, 0x30, 0x30, 0x30, 0x20, 0x31, 0x32, 0x3a, 0x34,
0x35,
        0x3a, 0x32, 0x38, 0x20, 0x47, 0x4d, 0x54, 0x0d, 0x0a, 0x43, 0x6f, 0x6e, 0x74, 0x65, 0x6e,
0x74,
        0x2d, 0x54, 0x79, 0x70, 0x65, 0x3a, 0x20, 0x74, 0x65, 0x78, 0x74, 0x2f, 0x68, 0x74, 0x6d,
0x6c,
        0x0d, 0x0a, 0x0d, 0x0a
ENDFIELD(UTCPPACKET_D0,152)

/* compressed TCP Paket: dummy packet E0
```

0xf8, 0x0c	SDU length (415 Byte = 3320 Bit)
0x00, 0x00	SDU offset
….	Data (415 byte)

```
*/
FIELD(CTCPPACKET_E0)
        0xf8, 0x0c,
        0x00, 0x00,
        0x3f, 0xbf, 0x67, 0x00, 0x01, 0x00, 0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x0d, 0x0a, 0x3c,
0x68,
        0x65, 0x61, 0x64, 0x3e, 0x0d, 0x0a, 0x0d, 0x0a, 0x3c, 0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e,
0x43,
        0x6f, 0x6e, 0x64, 0x61, 0x74, 0x3c, 0x2f, 0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x0d, 0x0a,
0x0d,
        0x0a, 0x0d, 0x0a, 0x3c, 0x73, 0x63, 0x72, 0x69, 0x70, 0x74, 0x20, 0x6c, 0x61, 0x6e, 0x67,
0x75,
        0x61, 0x67, 0x65, 0x3d, 0x22, 0x4a, 0x61, 0x76, 0x61, 0x53, 0x63, 0x72, 0x69, 0x70, 0x74,
0x22,
        0x3e, 0x20, 0x3c, 0x21, 0x2d, 0x2d, 0x0d, 0x0a, 0x0d, 0x0a, 0x0d, 0x0a, 0x66, 0x75, 0x6e,
0x63,
        0x74, 0x69, 0x6f, 0x6e, 0x20, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61,
0x67,
        0x65, 0x28, 0x29, 0x20, 0x7b, 0x20, 0x2f, 0x2f, 0x76, 0x31, 0x2e, 0x30, 0x0d, 0x0a, 0x20,
0x20,
        0x76, 0x61, 0x72, 0x20, 0x69, 0x2c, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a, 0x2c, 0x6a, 0x3d,
0x30,
        0x2c, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72, 0x61, 0x79, 0x3d, 0x6e, 0x65, 0x77, 0x20,
0x41,
        0x72, 0x72, 0x61, 0x79, 0x2c, 0x6f, 0x6c, 0x64, 0x41, 0x72, 0x72, 0x61, 0x79, 0x3d, 0x64,
0x6f,
        0x63, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x2e, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49,
0x6d,
        0x67, 0x44, 0x61, 0x74, 0x61, 0x3b, 0x0d, 0x0a, 0x20, 0x20, 0x66, 0x6f, 0x72, 0x20, 0x28,
0x69,
        0x3d, 0x30, 0x3b, 0x20, 0x69, 0x20, 0x3c, 0x20, 0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61,
0x70,
```

```
        0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e, 0x61, 0x72, 0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73,
0x2e,
        0x6c, 0x65, 0x6e, 0x67, 0x74, 0x68, 0x2d, 0x32, 0x29, 0x3b, 0x20, 0x69, 0x2b, 0x3d, 0x33,
0x29,
        0x20, 0x7b, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a, 0x20,
0x3d,
        0x20, 0x65, 0x76, 0x61, 0x6c, 0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d,
0x61,
        0x67, 0x65, 0x2e, 0x61, 0x72, 0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x5b, 0x28, 0x6e,
0x61,
        0x76, 0x69, 0x67, 0x61, 0x74, 0x6f, 0x72, 0x2e, 0x61, 0x70, 0x70, 0x4e, 0x61, 0x6d, 0x65,
0x20,
        0x3d, 0x3d, 0x20, 0x27, 0x4e, 0x65, 0x74, 0x73, 0x63, 0x61, 0x70, 0x65, 0x27, 0x29, 0x3f,
0x69,
        0x3a, 0x69, 0x2b, 0x31, 0x5d, 0x29, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x69, 0x66, 0x20,
0x28,
        0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a, 0x20, 0x21, 0x3d, 0x20, 0x6e, 0x75, 0x6c, 0x6c, 0x29,
0x20,
        0x7b, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72,
0x72,
        0x61, 0x79, 0x5b, 0x6a, 0x2b, 0x2b, 0x5d, 0x20, 0x3d, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62,
0x6a,
        0x3b, 0x0d, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72
ENDFIELD(CTCPPACKET_E0, 419)

/*
```

3.3.22 IP frames

```
*/
/* HDLC Frame: IP Packet A0
```

0xc0, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPFRAME_NF_A0)        0xc0, 0x03,
                            0x00, 0x00,
                            0x7e,
                            0xff,
                            0x7d, 0x23,
                            0x7d, 0x020, 0x21,
                            0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
```

```
                        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPFRAME_NF_A0, 124)

/* HDLC Frame: IP Packet A0 (ACFC, PFC)
```

0x98, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x21	0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPFRAME_NF_A0_C)    0x98, 0x03,
                        0x00, 0x00,
                        0x7e,
                        0x21,
                        0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPFRAME_NF_A0_C, 119)

/* HDLC Frame: IP Packet B0
```

0xc0, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPFRAME_NF_B0)      0xc0, 0x03,
                        0x00, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0x7d, 0x020, 0x21,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
```

```
                        0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPFRAME_NF_B0, 124)
```

/* HDLC Frame: IP Packet B0 (ACFC, PFC)

0x98, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x21	0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPFRAME_NF_B0_C)    0x98, 0x03,
                        0x00, 0x00,
                        0x7e,
                        0x21,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPFRAME_NF_B0_C, 119)
```

/* HDLC Frame: IP Packet C0

0xc0, 0x03		SDU length (120 Byte = 960 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xc1, 0xc2, 0xc3, …., 0xcf	0xc1, 0xc2, 0xc3, …., 0xcf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPFRAME_NF_C0)
        0xc0, 0x03,
        0x00, 0x00,
        0x7e,
```

```
        0xff,
        0x7d, 0x23,
        0x7d, 0x020, 0x21,
        0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xb1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xcf,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPFRAME_NF_C0, 124)

/* HDLC Frame: Three IP Packets in one Frame
```

0x38, 0x0b		SDU length (359 Byte = 2872 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2,…, 0xaf	0xa1, 0xa2,…, 0xaf	data (110 byte)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0x1b, 0xb2, …., 0xbf	0x1b, 0xb2, 0xb3, …., 0xbf	data (110 byte)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0x1c, 0xc2, …., 0xcf	0x1b, 0xb2, …., 0xcf	data (110 byte)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPFRAME_TRIPLE)
        0x38, 0x0b,
        0x00, 0x00,
        0x7e,
        0xff,
        0x7d, 0x23,
        0x7d, 0x020, 0x21,
        0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
```

```
        0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xf0, 0xb8,
        0x7e,
        0xff,
        0x7d, 0x23,
        0x7d, 0x020, 0x21,
        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
        0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
        0xf0, 0xb8,
        0x7e,
        0x7e,
        0xff,
        0x7d, 0x23,
        0x7d, 0x020, 0x21,
        0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xb1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
        0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xcf,
        0xf0, 0xb8,
        0x7e,
ENDFIELD(IPFRAME_TRIPLE, 363)
```

/* HDLC Frame: Three IP Packets in one Frame (ACFC, PFC)

0xc8, 0x0a		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x21	0x21	Protocol = IP
0xa1, 0xa2,…, 0xaf	0xa1, 0xa2,…, 0xaf	data (110 byte)
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag
0x7e		Flag
0x21	0x21	Protocol = IP
0x1b, 0xb2, …., 0xbf	0x1b, 0xb2, 0xb3, …., 0xbf	data (110 byte)

0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag
0x7e		Flag
0x21	0x21	Protocol = IP
0x1c, 0xc2, …., 0xcf	0x1b, 0xb2, …., 0xcf	data (110 byte)
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

*/

```
FIELD(IPFRAME_TRIPLE_C)   0xc8, 0x0a,
                          0x00, 0x00,
                          0x7e,
                          0x21,
                          0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                          0xf0, 0xb8,
                          0x7e,
                          0x7e,
                          0x21,
                          0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                          0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
                          0xf0, 0xb8,
                          0x7e,
                          0x7e,
                          0x21,
                          0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xb1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xd1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xe1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xf1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xc0,
                          0xa1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7, 0xc8, 0xc9, 0xcf,
                          0xf0, 0xb8,
                          0x7e,
ENDFIELD(IPFRAME_TRIPLE_C, 349)
```

/* HDLC Frame: uncompressed TCP (ACFC, PFC)

0xa0, 0x05		SDU length (180 Byte = 1440 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x00, 0x2f	0x2f	Protocol Field (Uncompressed TCP)
….		Data (175 Byte)
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

*/
FIELD(UTCPFRAME_NF_D0_C)
0xa0, 0x05,
0x00, 0x00,
0x7e,
0x2f,
0x45, 0x7d, 0x20, 0x7d, 0x20, 0x94, 0x7d, 0x24, 0x7d, 0x3e, 0x40, 0x7d, 0x20, 0x75, 0x7d, 0x20,
0xd9, 0x26, 0x3e, 0x60, 0xcf, 0x7d, 0x29, 0x97, 0xbd, 0x82, 0xf8, 0x7d, 0x20, 0x50, 0x7d, 0x24,
0x7d, 0x30, 0xe2, 0x75, 0xdd, 0x43, 0x7d, 0x20, 0x7d, 0x3a, 0xbe, 0x78, 0x50, 0x7d, 0x38, 0x21,
0x7d, 0x29, 0x7d, 0x38, 0xaf, 0x7d, 0x20, 0x7d, 0x20, 0x48, 0x54, 0x54, 0x50, 0x2f, 0x31, 0x2e,
0x31, 0x20, 0x32, 0x30, 0x30, 0x20, 0x4f, 0x4b, 0x7d, 0x2d, 0x7d, 0x2a, 0x53, 0x65, 0x72, 0x76,
0x65, 0x72, 0x3a, 0x20, 0x4d, 0x69, 0x63, 0x72, 0x6f, 0x73, 0x6f, 0x66, 0x74, 0x2d, 0x49, 0x49,
0x53, 0x2f, 0x34, 0x2e, 0x30, 0x7d, 0x2d, 0x7d, 0x2a, 0x44, 0x61, 0x74, 0x65, 0x3a, 0x20, 0x46,
0x72, 0x69, 0x2c, 0x20, 0x30, 0x36, 0x20, 0x4f, 0x63, 0x74, 0x20, 0x32, 0x30, 0x30, 0x30, 0x20,
0x31, 0x32, 0x3a, 0x34, 0x35, 0x3a, 0x32, 0x38, 0x20, 0x47, 0x4d, 0x54, 0x7d, 0x2d, 0x7d, 0x2a,
0x43, 0x6f, 0x6e, 0x74, 0x65, 0x6e, 0x74, 0x2d, 0x54, 0x79, 0x70, 0x65, 0x3a, 0x20, 0x74, 0x65,
0x78, 0x74, 0x2f, 0x68, 0x74, 0x6d, 0x6c, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a,
0xf0, 0xb8,
0x7e
ENDFIELD(UTCPFRAME_NF_D0_C, 184)

/* HDLC Frame: compressed TCP (ACFC, PFC)

0x28, 0x0e		SDU length (453 Byte = 3624 Bit)
0x00, 0x00		SDU offset
0x7e		Flag
0x00, 0x2d	0x2d	Protocol Field (Compressed TCP)
….		Data (448 Byte)
0xf0, 0xb8	0xf0, 0xb8	FCS (dummy)
0x7e		Flag

*/
FIELD(CTCPFRAME_NF_E0_C)
0x28, 0x0e,
0x00, 0x00,
0x7e,
0x2d,
0x3f, 0xbf, 0x67, 0x7d, 0x20, 0x7d, 0x21, 0x7d, 0x20, 0x3c, 0x68, 0x74, 0x6d, 0x6c, 0x3e, 0x7d,

```
        0x2d, 0x7d, 0x2a, 0x3c, 0x68, 0x65, 0x61, 0x64, 0x3e, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d,
0x7d,
        0x2a, 0x3c, 0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x43, 0x6f, 0x6e, 0x64, 0x61, 0x74, 0x3c,
0x2f,
        0x74, 0x69, 0x74, 0x6c, 0x65, 0x3e, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d,
0x2d,
        0x7d, 0x2a, 0x3c, 0x73, 0x63, 0x72, 0x69, 0x70, 0x74, 0x20, 0x6c, 0x61, 0x6e, 0x67, 0x75,
0x61,
        0x67, 0x65, 0x3d, 0x22, 0x4a, 0x61, 0x76, 0x61, 0x53, 0x63, 0x72, 0x69, 0x70, 0x74, 0x22,
0x3e,
        0x20, 0x3c, 0x21, 0x2d, 0x2d, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d, 0x7d, 0x2a, 0x7d, 0x2d,
0x7d,
        0x2a, 0x66, 0x75, 0x6e, 0x63, 0x74, 0x69, 0x6f, 0x6e, 0x20, 0x4d, 0x4d, 0x5f, 0x73, 0x77,
0x61,
        0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x28, 0x29, 0x20, 0x7b, 0x20, 0x2f, 0x2f, 0x76, 0x31,
0x2e,
        0x30, 0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x76, 0x61, 0x72, 0x20, 0x69, 0x2c, 0x74, 0x68,
0x65,
        0x4f, 0x62, 0x6a, 0x2c, 0x6a, 0x3d, 0x30, 0x2c, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72,
0x61,
        0x79, 0x3d, 0x6e, 0x65, 0x77, 0x20, 0x41, 0x72, 0x72, 0x61, 0x79, 0x2c, 0x6f, 0x6c, 0x64,
0x41,
        0x72, 0x72, 0x61, 0x79, 0x3d, 0x64, 0x6f, 0x63, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x2e, 0x4d,
0x4d,
        0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x67, 0x44, 0x61, 0x74, 0x61, 0x3b, 0x7d, 0x2d,
0x7d,
        0x2a, 0x20, 0x20, 0x66, 0x6f, 0x72, 0x20, 0x28, 0x69, 0x3d, 0x30, 0x3b, 0x20, 0x69, 0x20,
0x3c,
        0x20, 0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e,
0x61,
        0x72, 0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x2e, 0x6c, 0x65, 0x6e, 0x67, 0x74, 0x68,
0x2d,
        0x32, 0x29, 0x3b, 0x20, 0x69, 0x2b, 0x3d, 0x33, 0x29, 0x20, 0x7b, 0x7d, 0x2d, 0x7d, 0x2a,
0x20,
        0x20, 0x20, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a, 0x20, 0x3d, 0x20, 0x65, 0x76, 0x61,
0x6c,
        0x28, 0x4d, 0x4d, 0x5f, 0x73, 0x77, 0x61, 0x70, 0x49, 0x6d, 0x61, 0x67, 0x65, 0x2e, 0x61,
0x72,
        0x67, 0x75, 0x6d, 0x65, 0x6e, 0x74, 0x73, 0x5b, 0x28, 0x6e, 0x61, 0x76, 0x69, 0x67, 0x61,
0x74,
        0x6f, 0x72, 0x2e, 0x61, 0x70, 0x70, 0x4e, 0x61, 0x6d, 0x65, 0x20, 0x3d, 0x3d, 0x20, 0x27,
0x4e,
        0x65, 0x74, 0x73, 0x63, 0x61, 0x70, 0x65, 0x27, 0x29, 0x3f, 0x69, 0x3a, 0x69, 0x2b, 0x31,
0x5d,
        0x29, 0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x69, 0x66, 0x20, 0x28, 0x74, 0x68,
0x65,
        0x4f, 0x62, 0x6a, 0x20, 0x21, 0x3d, 0x20, 0x6e, 0x75, 0x6c, 0x6c, 0x29, 0x20, 0x7b, 0x7d,
0x2d,
        0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41, 0x72, 0x72,
0x61,
        0x79, 0x5b, 0x6a, 0x2b, 0x2b, 0x5d, 0x20, 0x3d, 0x20, 0x74, 0x68, 0x65, 0x4f, 0x62, 0x6a,
0x3b,
        0x7d, 0x2d, 0x7d, 0x2a, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x73, 0x77, 0x61, 0x70, 0x41,
0x72,
        0xf0, 0xb8,
        0x7e
ENDFIELD(CTCPFRAME_NF_E0_C, 457)

/*
```

3.3.23 HDLC fragments

*/
/* HDLC Frame: IP Packet A0 Fragment 1 (1/2)

0x18, 0x02		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, ….	0xa1, 0xa2, 0xa3, …., 0xaf	data

*/

```
FIELD(IPFRAME_F_A0F1)       0x18, 0x02,
                            0x00, 0x00,
                            0x7e,
                            0xff,
                            0x7d, 0x23,
                            0x7d, 0x020, 0x21,
                            0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                            0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
ENDFIELD(IPFRAME_F_A0F1, 71)
```

/* HDLC Frame: IP Packet A0 Fragment 1 (1/2) (ACFC, PFC)

0xf0, 0x01		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x21	0x21	Protocol = IP
0xa1, 0xa2, 0xa3, ….	0xa1, 0xa2, 0xa3, …., 0xaf	data

*/

```
FIELD(IPFRAME_F_A0F1_C)
        0xf0, 0x01,
        0x00, 0x00,
        0x7e,
        0x21,
        0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
ENDFIELD(IPFRAME_F_A0F1_C, 66)
```

/* HDLC Frame: IP Packet A0 Fragment 2 (2/2)

0xa8, 0x01		SDU length
0x00, 0x00		SDU offset
…., 0xaf	…., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/

```
FIELD(IPFRAME_F_A0F2)       0xa8, 0x01,
                            0x00, 0x00,
                            0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
```

```
                        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPFRAME_F_A0F2, 57)
```

/* HDLC Frame: IP Packet A0 Fragment 2 (2/2)) (ACFC, PFC)

0xa8, 0x01		SDU length
0x00, 0x00		SDU offset
…., 0xaf	…., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(IPFRAME_F_A0F2_C)
        0xa8, 0x01,
        0x00, 0x00,
        0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
        0xf0, 0xb8,
        0x7e
ENDFIELD(IPFRAME_F_A0F2_C, 57)
```

/* HDLC Frame: IP Packet A0 Fragment 2 (2/2) and IP Packet B0 Framgment 1 (1/2)

0xb8, 0x03		SDU length
0x00, 0x00		SDU offset
…., 0xaf	…., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0x1b, 0xb2, 0xb3, ….	0x1b, 0xb2, 0xb3, ….	data

```
*/
FIELD(IPFRAME_F_A0F2B0F1)       0xb8, 0x03,
                        0x00, 0x00,
                        0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf0, 0xb8,
                        0x7e,
                        0xff,
                        0x7d, 0x23,
                        0x7d, 0x020, 0x21,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
```

ENDFIELD(IPFRAME_F_A0F2B0F1, 123)

/* HDLC Frame: IP Packet A0 Fragment 2 (2/2) and IP Packet B0 Framgment 1 (1/2) (ACFC, PFC)

0x90, 0x03		SDU length
0x00, 0x00		SDU offset
…., 0xaf	…., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag
0x21	0x21	Protocol = IP
0x1b, 0xb2, 0xb3, ….	0x1b, 0xb2, 0xb3, ….	data

*/
FIELD(IPFRAME_F_A0F2B0F1_C)
0x90, 0x03,
0x00, 0x00,
0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xf0, 0xb8,
0x7e,
0x21,
0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x2b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x3b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x4b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x5b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x6b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
ENDFIELD(IPFRAME_F_A0F2B0F1_C, 118)

/* HDLC Frame: IP Packet A0 Fragment 2 (2/2)

0xa8, 0x01		SDU length
0x00, 0x00		SDU offset
…., 0xaf	…., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPFRAME_F_B0F2) 0xa8, 0x01,
0x00, 0x00,
0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
0xf0, 0xb8,
0x7e
ENDFIELD(IPFRAME_F_B0F2, 57)

/* HDLC Frame: IP Packet A0 Fragment 2 (2/2) (ACFC, PFC)

0xa8, 0x01		SDU length
0x00, 0x00		SDU offset
…., 0xaf	…., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(IPFRAME_F_B0F2_C) 0xa8, 0x01,

```
                        0x00, 0x00,
                        0x7b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x8b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0x9b, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfb, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xb0,
                        0xfc, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7, 0xb8, 0xb9, 0xbf,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(IPFRAME_F_B0F2_C, 57)

/*
```

3.3.24 Invalid frames

```
*/
/* IP Packet in HDLC Frame (invalid Control Field)
```

0xc0, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x04	0x7d, 0x24	Control Field (invalid!!!)
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

```
*/
FIELD(INVALID_FRAME0) 0xc0, 0x03,
                        0x00, 0x00,
                        0x7e,
                        0xff,
                        0x7d, 0x24,
                        0x7d, 0x020, 0x21,
                        0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                        0xf0, 0xb8,
                        0x7e
ENDFIELD(INVALID_FRAME0, 124)

/* IP Packet in HDLC Frame (invalid protocol field)
```

0xb8, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0x08	0x08	Address Field (invalid protocol field!!!)
0x20	0x20	Control Field (invalid protocol field!!!)
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, ….,	0xa1, 0xa2, 0xa3, …., 0xaf	data

0xaf		
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(INVALID_FRAME1) 0xb8, 0x03,
0x00, 0x00,
0x7e,
0x08,
0x20,
0x7d, 0x020, 0x21,
0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xf0, 0xb8,
0x7e
ENDFIELD(INVALID_FRAME1, 123)

/* IP Packet in HDLC Frame (invalid address Field)

0xc0, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xf7	0xf7	Address Field (invalid!!!)
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xb8	0xf0, 0xb8	FSC (dummy)
0x7e		Flag

*/
FIELD(INVALID_FRAME2) 0xc0, 0x03,
0x00, 0x00,
0x7e,
0xf7,
0x7d, 0x23,
0x7d, 0x020, 0x21,
0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
0xf0, 0xb8,
0x7e
ENDFIELD(INVALID_FRAME2, 124)

/* IP Packet in HDLC Frame (invalid FSC)

0xc0, 0x03		SDU length
0x00, 0x00		SDU offset
0x7e		Flag
0xff	0xff	Address Field
0x03	0x7d, 0x23	Control Field
0x00, 0x21	0x7d, 0x020, 0x21	Protocol = IP
0xa1, 0xa2, 0xa3, …., 0xaf	0xa1, 0xa2, 0xa3, …., 0xaf	data
0xf0, 0xc8	0xf0, 0xc8	FSC (dummy) (invalid!!!)
0x7e		Flag

*/
```
FIELD(INVALID_FSC) 0xc0, 0x03,
                   0x00, 0x00,
                   0x7e,
                   0xff,
                   0x7d, 0x23,
                   0x7d, 0x020, 0x21,
                   0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xb1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xc1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xd1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xe1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xf1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xa2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xb2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xc2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xd2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xe2, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7, 0xa8, 0xa9, 0xa0,
                   0xf0, 0xc8,
                   0x7e
ENDFIELD(INVALID_FSC, 124)

/*
```

3.3.25 Empty sdu

*/
/* empty sdu

0x00, 0x00	SDU length
0x00, 0x00	SDU offset

*/
```
FIELD(EMPTY_SDU)
        0x00, 0x00,
        0x00, 0x00
ENDFIELD(EMPTY_SDU, 4)

/*
```

3.3.26 PCO lists

*/
/* Protocol Configuration Options for PPP_PDP_ACTIVATE_IND with CHAP

0x98, 0x03	SDU length in bit (115 Byte = 920 Bit)
0x00, 0x00	SDU offset in bit
0x80	Protocol PPP see GSM04.08 page 569

0xc0, 0x21	Protocol ID1: LCP
0x0d	Length of ID1: 13
0x01, 0x01, 0x00, 0x0d	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0x03, 0x05, 0xc2, 0x23, 0x05	Authentication Protocol CHAP with MD5
0xc0, 0x21	Protocol ID2: LCP
0x08	Length of ID2: 8
0x01, 0x01, 0x00, 0x08	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0xc2, 0x23	Protocol ID3: CHAP
0x15	Length of ID3: 21
0x01, 0x01, 0x00, 0x15	Code, Identifier and Length Field: Challenge
0x10, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f	Content of Challenge
0xc2, 0x23	Protocol ID4: CHAP
0x1d	Length of ID4: 29
0x02, 0x01, 0x00, 0x1d	Code, Identifier and Length Field: Response
0x10, 0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3, 0x13, 0x08, 0xc2, 0x2f, 0xd7, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	Content of Response
0x80, 0x21	Protocol ID5: IPCP
0x1c	Length of ID5: 28
0x01, 0x01, 0x00, 0x1c	Code, Identifier and Length Field: Configure-Request
0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01	Van Jacobson Header Compression
0x03, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic IP-Address
0x81, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Primary DNS Address
0x83, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Secondary DNS Address

```
*/
FIELD(PCO_LIST_IND_CHAP)
        0x98, 0x03,
        0x00, 0x00,
        0x80,
        0xc0, 0x21, 0x0d,
        0x01, 0x01, 0x00, 0x0d, 0x01, 0x04, 0x05, 0xdc, 0x03, 0x05, 0xc2, 0x23, 0x05,
        0xc0, 0x21, 0x08,
        0x01, 0x01, 0x00, 0x08, 0x01, 0x04, 0x05, 0xdc,
        0xc2, 0x23, 0x15,
        0x01, 0x01, 0x00, 0x15, 0x10, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,
0x0a,
        0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
        0xc2, 0x23, 0x1d,
        0x02, 0x01, 0x00, 0x1d, 0x10, 0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4,
0xe3,
```

```
        0x13, 0x08, 0xc2, 0x2f, 0xd7, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65,
        0x80, 0x21, 0x1c,
        0x01, 0x01, 0x00, 0x1c, 0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01, 0x03, 0x06, 0x00, 0x00, 0x00,
0x00,
        0x81, 0x06, 0x00, 0x00, 0x00, 0x00, 0x83, 0x06, 0x00, 0x00, 0x00, 0x00
ENDFIELD(PCO_LIST_IND_CHAP, 119)

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_IND with PAP
```

0x98, 0x02	SDU length in bit (83 Byte = 664 Bit)
0x00, 0x00	SDU offset in bit
0x80	Protocol PPP see GSM04.08 page 569
0xc0, 0x21	Protocol ID1: LCP
0x0c	Length of ID1: 12
0x01, 0x01, 0x00, 0x0c	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0x03, 0x04, 0xc0, 0x23	Authentication Protocol PAP
0xc0, 0x21	Protocol ID2: LCP
0x08	Length of ID2: 8
0x01, 0x01, 0x00, 0x08	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0xc0, 0x23	Protocol ID3: PAP
0x16	Length of ID3: 22
0x01, 0x00, 0x00, 0x16	Code, Identifier and Length Field: Authenticate-Request
0x08, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	Length and Peer-ID
0x08, 0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	Length and Password
0x80, 0x21	Protocol ID4: IPCP
0x1c	Length of ID4: 28
0x01, 0x01, 0x00, 0x1c	Code, Identifier and Length Field: Configure-Request
0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01	Van Jacobson Header Compression
0x03, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic IP-Address
0x81, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Primary DNS Address
0x83, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Secondary DNS Address

```
*/
FIELD(PCO_LIST_IND_PAP)
        0x98, 0x02,
        0x00, 0x00,
        0x80,
        0xc0, 0x21, 0x0c,
        0x01, 0x01, 0x00, 0x0c, 0x01, 0x04, 0x05, 0xdc, 0x03, 0x04, 0xc0, 0x23,
        0xc0, 0x21, 0x08,
        0x01, 0x01, 0x00, 0x08, 0x01, 0x04, 0x05, 0xdc,
        0xc0, 0x23, 0x16,
        0x01, 0x00, 0x00, 0x16, 0x08, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65, 0x08, 0x74,
        0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73,
        0x80, 0x21, 0x1c,
        0x01, 0x01, 0x00, 0x1c, 0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01, 0x03, 0x06, 0x00, 0x00, 0x00,
0x00,
        0x81, 0x06, 0x00, 0x00, 0x00, 0x00, 0x83, 0x06, 0x00, 0x00, 0x00, 0x00
```

ENDFIELD(PCO_LIST_IND_PAP, 87)

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_IND without authentication

0xb0, 0x01	SDU length in bit (54 Byte = 432 Bit)
0x00, 0x00	SDU offset in bit
0x80	Protocol PPP see GSM04.08 page 569
0xc0, 0x21	Protocol ID1: LCP
0x08	Length of ID1: 12
0x01, 0x01, 0x00, 0x08	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0xc0, 0x21	Protocol ID2: LCP
0x08	Length of ID2: 8
0x01, 0x01, 0x00, 0x08	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0x80, 0x21	Protocol ID4: IPCP
0x1c	Length of ID4: 28
0x01, 0x01, 0x00, 0x1c	Code, Identifier and Length Field: Configure-Request
0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01	Van Jacobson Header Compression
0x03, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic IP-Address
0x81, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Primary DNS Address
0x83, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Secondary DNS Address

*/
FIELD(PCO_LIST_IND_NO)
0xb0, 0x01,
0x00, 0x00,
0x80,
0xc0, 0x21, 0x08,
0x01, 0x01, 0x00, 0x08, 0x01, 0x04, 0x05, 0xdc,
0xc0, 0x21, 0x08,
0x01, 0x01, 0x00, 0x08, 0x01, 0x04, 0x05, 0xdc,
0x80, 0x21, 0x1c,
0x01, 0x01, 0x00, 0x1c, 0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01, 0x03, 0x06, 0x00, 0x00, 0x00, 0x00,
0x81, 0x06, 0x00, 0x00, 0x00, 0x00, 0x83, 0x06, 0x00, 0x00, 0x00, 0x00
ENDFIELD(PCO_LIST_IND_NO, 58)

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_IND with CHAP

0x68, 0x03	SDU length in bit (109 Byte = 872 Bit)
0x00, 0x00	SDU offset in bit
0x80	Protocol PPP see GSM04.08 page 569
0xc0, 0x21	Protocol ID1: LCP
0x0d	Length of ID1: 13
0x01, 0x01, 0x00, 0x0d	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0x03, 0x05, 0xc2, 0x23, 0x05	Authentication Protocol CHAP with MD5
0xc0, 0x21	Protocol ID2: LCP
0x08	Length of ID2: 8
0x01, 0x01, 0x00, 0x08	Code, Identifier and Length Field: Configure-Request

0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0xc2, 0x23	Protocol ID3: CHAP
0x15	Length of ID3: 21
0x01, 0x01, 0x00, 0x15	Code, Identifier and Length Field: Challenge
0x10, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f	Content of Challenge
0xc2, 0x23	Protocol ID4: CHAP
0x1d	Length of ID4: 29
0x02, 0x01, 0x00, 0x1d	Code, Identifier and Length Field: Response
0x10, 0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3, 0x13, 0x08, 0xc2, 0x2f, 0xd7, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	Content of Response
0x80, 0x21	Protocol ID5: IPCP
0x16	Length of ID5: 22
0x01, 0x01, 0x00, 0x16	Code, Identifier and Length Field: Configure-Request
0x03, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic IP-Address
0x81, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Primary DNS Address
0x83, 0x06, 0x00, 0x00, 0x00, 0x00	Dynamic Secondary DNS Address

*/
FIELD(PCO_LIST_IND_CHAP_NOVJ)
0x68, 0x03,
0x00, 0x00,
0x80,
0xc0, 0x21, 0x0d,
0x01, 0x01, 0x00, 0x0d, 0x01, 0x04, 0x05, 0xdc, 0x03, 0x05, 0xc2, 0x23, 0x05,
0xc0, 0x21, 0x08,
0x01, 0x01, 0x00, 0x08, 0x01, 0x04, 0x05, 0xdc,
0xc2, 0x23, 0x15,
0x01, 0x01, 0x00, 0x15, 0x10, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a,
0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
0xc2, 0x23, 0x1d,
0x02, 0x01, 0x00, 0x1d, 0x10, 0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3,
0x13, 0x08, 0xc2, 0x2f, 0xd7, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65,
0x80, 0x21, 0x16,
0x01, 0x01, 0x00, 0x16, 0x03, 0x06, 0x00, 0x00, 0x00, 0x00, 0x81, 0x06, 0x00, 0x00, 0x00, 0x00,
0x83, 0x06, 0x00, 0x00, 0x00, 0x00
ENDFIELD(PCO_LIST_IND_CHAP_NOVJ, 113)

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_IND with CHAP and static IP and DNS addresses

0x98, 0x03	SDU length in bit (115 Byte = 920 Bit)
0x00, 0x00	SDU offset in bit
0x80	Protocol PPP see GSM04.08 page 569

0xc0, 0x21	Protocol ID1: LCP
0x0d	Length of ID1: 13
0x01, 0x01, 0x00, 0x0d	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0x03, 0x05, 0xc2, 0x23, 0x05	Authentication Protocol CHAP with MD5
0xc0, 0x21	Protocol ID2: LCP
0x08	Length of ID2: 8
0x01, 0x01, 0x00, 0x08	Code, Identifier and Length Field: Configure-Request
0x01, 0x04, 0x05, 0xdc	MRU 1500 octets
0xc2, 0x23	Protocol ID3: CHAP
0x15	Length of ID3: 21
0x01, 0x01, 0x00, 0x15	Code, Identifier and Length Field: Challenge
0x10, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f	Content of Challenge
0xc2, 0x23	Protocol ID4: CHAP
0x1d	Length of ID4: 29
0x02, 0x01, 0x00, 0x1d	Code, Identifier and Length Field: Response
0x10, 0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4, 0xe3, 0x13, 0x08, 0xc2, 0x2f, 0xd7, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	Content of Response
0x80, 0x21	Protocol ID5: IPCP
0x1c	Length of ID5: 28
0x01, 0x01, 0x00, 0x1c	Code, Identifier and Length Field: Configure-Request
0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01	Van Jacobson Header Compression
0x03, 0x06, 0x8d, 0x40, 0x15, 0x02	IP-Address 141.64.21.2
0x81, 0x06, 0x8d, 0x40, 0x18, 0x81	Primary DNS Address 141.64.24.129
0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83	Secondary DNS Address 141.64.254.131

*/

```
FIELD(PCO_LIST_IND_CHAP_STATIC)
        0x98, 0x03,
        0x00, 0x00,
        0x80,
        0xc0, 0x21, 0x0d,
        0x01, 0x01, 0x00, 0x0d, 0x01, 0x04, 0x05, 0xdc, 0x03, 0x05, 0xc2, 0x23, 0x05,
        0xc0, 0x21, 0x08,
        0x01, 0x01, 0x00, 0x08, 0x01, 0x04, 0x05, 0xdc,
        0xc2, 0x23, 0x15,
        0x01, 0x01, 0x00, 0x15, 0x10, 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09,
0x0a,
        0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
        0xc2, 0x23, 0x1d,
        0x02, 0x01, 0x00, 0x1d, 0x10, 0xd1, 0xe3, 0x7f, 0x02, 0x2a, 0x0c, 0xcd, 0x13, 0x85, 0xd4,
0xe3,
```

```
        0x13, 0x08, 0xc2, 0x2f, 0xd7, 0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65,
        0x80, 0x21, 0x1c,
        0x01, 0x01, 0x00, 0x1c, 0x02, 0x06, 0x00, 0x2d, 0x0f, 0x01, 0x03, 0x06, 0x8d, 0x40, 0x15,
0x02,
        0x81, 0x06, 0x8d, 0x40, 0x18, 0x81, 0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83
ENDFIELD(PCO_LIST_IND_CHAP_STATIC, 119)
```

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_RES

0xd0, 0x00	SDU length in bit (26 Byte = 208 Bit)
0x00, 0x00	SDU offset in bit
0x80	Protocol PPP see GSM04.08 page 569
0x80, 0x21	Protocol ID1: IPCP
0x16	Length of ID1: 22
0x03, 0x01, 0x00, 0x16	Code, Identifier and Length Field: Configure-NAK
0x03, 0x06, 0x8d, 0x40, 0x66, 0x03	IP-Address 141.64.102.3
0x81, 0x06, 0x8d, 0x40, 0x18, 0x81	Primary DNS Address 141.64.24.129
0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83	Secondary DNS Address 141.64.254.131

```
*/
FIELD(PCO_LIST_RES)
        0xd0, 0x00,
        0x00, 0x00,
        0x80,
        0x80, 0x21, 0x16,
        0x03, 0x01, 0x00, 0x16, 0x03, 0x06, 0x8d, 0x40, 0x66, 0x03, 0x81, 0x06, 0x8d, 0x40, 0x18,
0x81,
        0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83
ENDFIELD(PCO_LIST_RES, 30)
```

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_RES with sdu offset

0xd0, 0x00	SDU length in bit (26 Byte = 208 Bit)
0x08, 0x00	SDU offset in bit (1 Byte = 8 Bit)
0x00	Offset octet
0x80	Protocol PPP see GSM04.08 page 569
0x80, 0x21	Protocol ID1: IPCP
0x16	Length of ID1: 22
0x03, 0x01, 0x00, 0x16	Code, Identifier and Length Field: Configure-NAK
0x03, 0x06, 0x8d, 0x40, 0x66, 0x03	IP-Address 141.64.102.3
0x81, 0x06, 0x8d, 0x40, 0x18, 0x81	Primary DNS Address 141.64.24.129
0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83	Secondary DNS Address 141.64.254.131

```
*/
FIELD(PCO_LIST_RES_OFFSET)
        0xd0, 0x00,
        0x08, 0x00,
        0x00,
        0x80,
        0x80, 0x21, 0x16,
        0x03, 0x01, 0x00, 0x16, 0x03, 0x06, 0x8d, 0x40, 0x66, 0x03, 0x81, 0x06, 0x8d, 0x40, 0x18,
0x81,
        0x83, 0x06, 0x8d, 0x40, 0xfe, 0x83
ENDFIELD(PCO_LIST_RES_OFFSET, 31)
```

/* Protocol Configuration Options for PPP_PDP_ACTIVATE_RES empty list

0x00, 0x00	SDU length in bit (0 Byte = 0 Bit)
0x00, 0x00	SDU offset in bit

*/
FIELD(PCO_LIST_RES_EMPTY)
0x00, 0x00,
0x00, 0x00,
ENDFIELD(PCO_LIST_RES_EMPTY, 4)

/*

3.4 Declarations

*/
DECLARATION(AUTH_STRUCT0)
DECLARATION (LOGIN_NAME)
DECLARATION (LOGIN_PASSWORD)

DECLARATION (PEER_CHANNEL0)
DECLARATION (PEER_CHANNEL1)
DECLARATION (PROTOCOL_CHANNEL0)
DECLARATION (PROTOCOL_CHANNEL1)

DECLARATION (DTI_CHANNELNAME_UART)
DECLARATION (DTI_CHANNELNAME_IP)
DECLARATION (DTI_CHANNELNAME_L2R)
DECLARATION (DTI_CHANNELNAME_MMI)

DECLARATION (SMREG_QOS_DUMMY)
DECLARATION (SMREG_MIN_QOS_DUMMY)
DECLARATION (PDP_ADDRESS_DUMMY)
DECLARATION (PDP_ADDRESS_BUF_DUMMY)
DECLARATION (SMREG_APN_DUMMY)
DECLARATION (SMREG_APN_BUF_DUMMY)
DECLARATION (DTI_PARAMETER_PEER)
DECLARATION (DTI_PARAMETER_PROT)
DECLARATION (DTI_PARAMETER_PROT2)
DECLARATION (DTI_PARAMETER_PROT3)
DECLARATION (DTI_PARA_ST_LINES_1)

/*

3.5 Arrays

*/

/* Login Name

0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65	name value ("test-name")

*/
BEGINARRAY(LOGIN_NAME, LOGIN_NAME_LEN)
0x74, 0x65, 0x73, 0x74, 0x6e, 0x61, 0x6d, 0x65
ENDARRAY

/* Login Password

0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73	name value ("testpass")

*/

```
BEGINARRAY(LOGIN_PASSWORD, LOGIN_PASSWORD_LEN)
                0x74, 0x65, 0x73, 0x74, 0x70, 0x61, 0x73, 0x73
ENDARRAY

/* DTI Communication Channel Name MMI
```

0x4d, 0x4d, 0x49, 0x00, 0x00, 0x0	name value ("MMI\0")

```
*/
BEGINARRAY(DTI_CHANNELNAME_MMI, 6)
                0x4d, 0x4d, 0x49, 0x00, 0x00, 0x00
ENDARRAY

/* DTI Communication Channel Name UART
```

0x55, 0x41, 0x52, 0x54, 0x00, 0x0	name value ("UART\0")

```
*/
BEGINARRAY(DTI_CHANNELNAME_UART, 6)
                0x55, 0x41, 0x52, 0x54, 0x00, 0x00
ENDARRAY

/* DTI Communication Channel Name IP
```

0x55, 0x41, 0x52, 0x54, 0x00, 0x0	name value ("IP\0")

```
*/
BEGINARRAY(DTI_CHANNELNAME_IP, 6)
                0x49, 0x50, 0x00, 0x00, 0x00, 0x00
ENDARRAY

/* DTI Communication Channel Name L2R
```

0x55, 0x41, 0x52, 0x54, 0x00, 0x0	name value ("L2R\0")

```
*/
BEGINARRAY(DTI_CHANNELNAME_L2R, 6)
                0x4C, 0x32, 0x52, 0x00, 0x00, 0x00
ENDARRAY

/* PDP address dummy
```

0x8d, 0x40, 0x15, 0x02	PDP address 141.64.21.2

```
*/
BEGINARRAY(PDP_ADDRESS_BUF_DUMMY, 4)
                0x8d, 0x40, 0x15, 0x02
ENDARRAY

/* smreg access point name dummy
```

0x41, 0x50, 0x4e	name value ("APN\0")

```
*/
BEGINARRAY(SMREG_APN_BUF_DUMMY, 3)
                0x41, 0x50, 0x4e
ENDARRAY

/*
```

3.6 Structs

```
*/

BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_1)
        SET_COMP ("st_flow",    DTI_FLOW_ON)
         SET_COMP ("st_line_sa",        DTI_SA_ON)
         SET_COMP ("st_line_sb",        DTI_SB_ON)
```

```
        SET_COMP ("st_break_len", DTI_BREAK_OFF)
ENDSTRUCT

BEGIN_PSTRUCT ("parameters",  DTI_PARAMETER_PEER)
        SET_COMP ("p_id",      DTI_PID_UOS)
        SET_COMP ("st_lines", DTI_PARA_ST_LINES_1)
ENDSTRUCT

BEGIN_PSTRUCT ("parameters",  DTI_PARAMETER_PROT)
        SET_COMP ("p_id",      DTI_PID_IP)
        SET_COMP ("st_lines", DTI_PARA_ST_LINES_1)
ENDSTRUCT


BEGIN_PSTRUCT ("parameters",  DTI_PARAMETER_PROT2)
        SET_COMP ("p_id",      DTI_PID_UTCP)
        SET_COMP ("st_lines", DTI_PARA_ST_LINES_1)
ENDSTRUCT

BEGIN_PSTRUCT ("parameters",  DTI_PARAMETER_PROT3)
        SET_COMP ("p_id",      DTI_PID_CTCP)
        SET_COMP ("st_lines", DTI_PARA_ST_LINES_1)
ENDSTRUCT

BEGIN_PSTRUCT("login", AUTH_STRUCT0)
        SET_COMP("name_len", LOGIN_NAME_LEN)
        SET_COMP("name", LOGIN_NAME)
        SET_COMP("password_len", LOGIN_PASSWORD_LEN)
        SET_COMP("password", LOGIN_PASSWORD)
ENDSTRUCT

BEGIN_PSTRUCT("peer_channel", PEER_CHANNEL0)
        SET_COMP("peer_entity", DTI_CHANNELNAME_UART)
ENDSTRUCT

BEGIN_PSTRUCT("peer_channel", PEER_CHANNEL1)
        SET_COMP("peer_entity", DTI_CHANNELNAME_L2R)
ENDSTRUCT

BEGIN_PSTRUCT("protocol_channel", PROTOCOL_CHANNEL0)
        SET_COMP("protocol_entity", DTI_CHANNELNAME_MMI)
ENDSTRUCT

BEGIN_PSTRUCT("protocol_channel", PROTOCOL_CHANNEL1)
        SET_COMP("protocol_entity", DTI_CHANNELNAME_IP)
ENDSTRUCT
```

4 TEST CASES

4.1 Setup (PPP001 - PPP010)

4.1.1 PPP001: Setup

Description:

Initialization of TAP

Preamble:

None

```
MMI / HIGHER LAYER              PPP                        LOWER LAYER
      |                          |                              |
COMMAND (TAP RESET)
COMMAND (MMI RESET)
COMMAND (UART RESET)
COMMAND (PPP RESET)
      |                          |                              |
COMMAND (TAP REDIRECT CLEAR)
COMMAND (MMI REDIRECT CLEAR)
COMMAND (UART REDIRECT CLEAR)
COMMAND (PPP REDIRECT CLEAR)
      |                          |                              |
COMMAND (PPP REDIRECT UART TAP)
COMMAND (PPP REDIRECT MMI TAP)
      |                          |                              |
COMMAND (TAP REDIRECT TAP PPP)
      |                          |                              |
```

Parametrization:

Primitive	Parameter	Value

History:

17-Jun-2000	SG	Initial

4.2 Link Establishment Requests (PPP011 – PPP019)

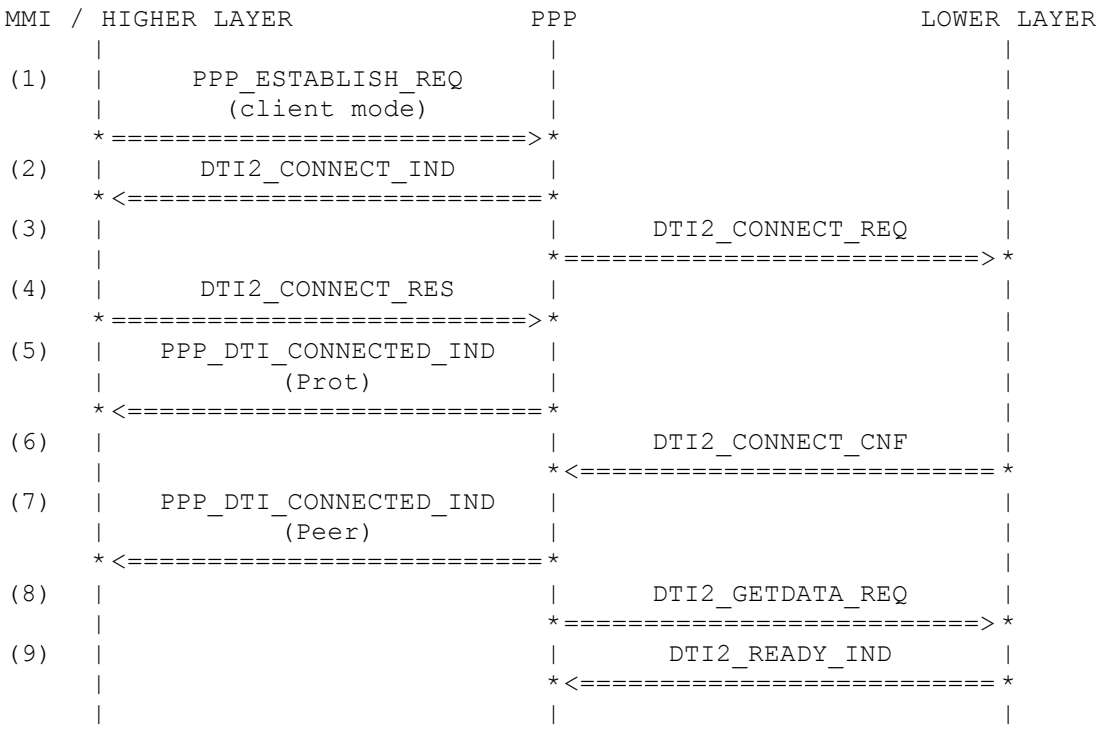
4.2.1 PPP011: Link establishment in client mode (default parameters)

Description:

Establishment in client mode whithout any authentification.

Preamble:

PPP001



Parametrization:

Primitive	Parameter	Value
(1) PPP_ESTABLISH_REQ		
	mode	PPP_CLIENT
	mru	PPP_MRU_DEFAULT
	ap	PPP_AP_NO
	login	AUTH_STRUCT0
	accm	PPP_ACCM_DEFAULT
	rt	PPP_RT_DEFAULT
	mc	PPP_MC_DEFAULT
	mt	PPP_MT_DEFAULT
	mf	PPP_MF_DEFAULT
	ppp_hc	PPP_HC_OFF
	ip	PPP_IP_DYNAMIC
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
	peer_channel	PEER_CHANNEL0
	protocol_channel	PROTOCOL_CHANNEL0
	peer_direction	DTI_CHANNEL_TO_LOWER_LAYER
	prot_direction	DTI_CHANNEL_TO_HIGHER_LAYER
	peer_link_id	LINK_ID_PEER
	prot_link_id	LINK_ID_PROT
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(3) DTI2_CONNECT_REQ		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(4) DTI2_CONNECT_RES		
	link_id	LINK_ID_PROT

	version	DTI_VERSION_10
(5) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PROT
(6) DTI2_CONNECT_CNF		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(7) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PEER
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(9) DTI2_READY_IND		
	link_id	LINK_ID_PEER

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID and connect primitive added.
03-Juli-2001	XOF	Position of Synchronisation Primitive changed
12-Jul-2001	STW	remove NOT_USED values
13-Sept-2001	XOF	Build DTILIB functionality

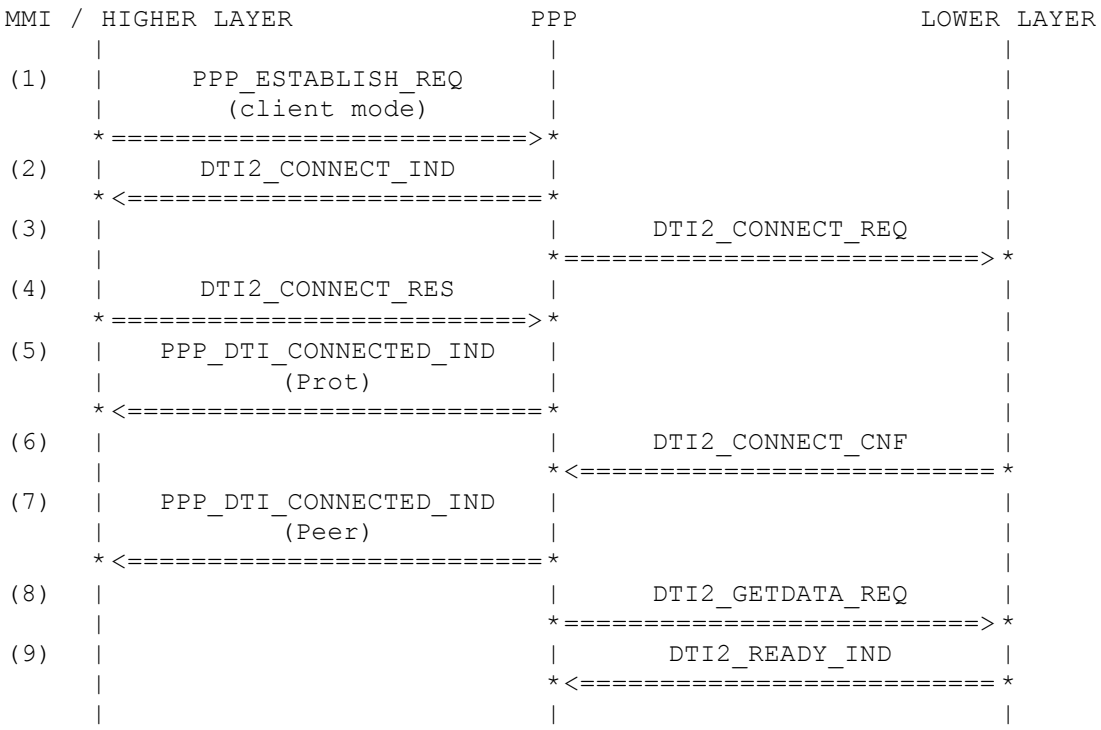
4.2.2 PPP012: Link establishment in client mode (PAP)

Description:

Establishment in client mode whithout any authentification.

Preamble:

PPP001



Parametrization:

Primitive	Parameter	Value
(1) PPP_ESTABLISH_REQ		
	mode	PPP_CLIENT
	mru	PPP_MRU_DEFAULT
	ap	PPP_AP_PAP
	login	AUTH_STRUCT0
	accm	PPP_ACCM_DEFAULT
	rt	PPP_RT_DEFAULT
	mc	PPP_MC_DEFAULT
	mt	PPP_MT_DEFAULT
	mf	PPP_MF_DEFAULT
	ppp_hc	PPP_HC_OFF
	ip	PPP_IP_DYNAMIC
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
	peer_channel	PEER_CHANNEL0
	protocol_channel	PROTOCOL_CHANNEL0
	peer_direction	DTI_CHANNEL_TO_LOWER_LAYER
	prot_direction	DTI_CHANNEL_TO_HIGHER_LAYER
	peer_link_id	LINK_ID_PEER
	prot_link_id	LINK_ID_PROT
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(3) DTI2_CONNECT_REQ		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10

(4)	DTI2_CONNECT_RES		
		link_id	LINK_ID_PROT
		version	
		DTI_VERSION_10	
(5)	PPP_DTI_CONNECTED_IND		
		connected_direction	PPP_DTI_CONN_PROT
(6)	DTI2_CONNECT_CNF		
		link_id	LINK_ID_PEER
		version	
		DTI_VERSION_10	
(7)	PPP_DTI_CONNECTED_IND		
		connected_direction	PPP_DTI_CONN_PEER
(8)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(9)	DTI2_READY_IND		
		link_id	LINK_ID_PEER

History:

17-Jun-2000	SG	Initial
22-Jun-2001	XOF	Channel ID and connect primitive added.
12-Jul-2001	STW	Remove NOT_USED values.
13-Sept-2001	XOF	Build DTILIB functionality

4.2.3 PPP013: Link establishment in server mode

Description:

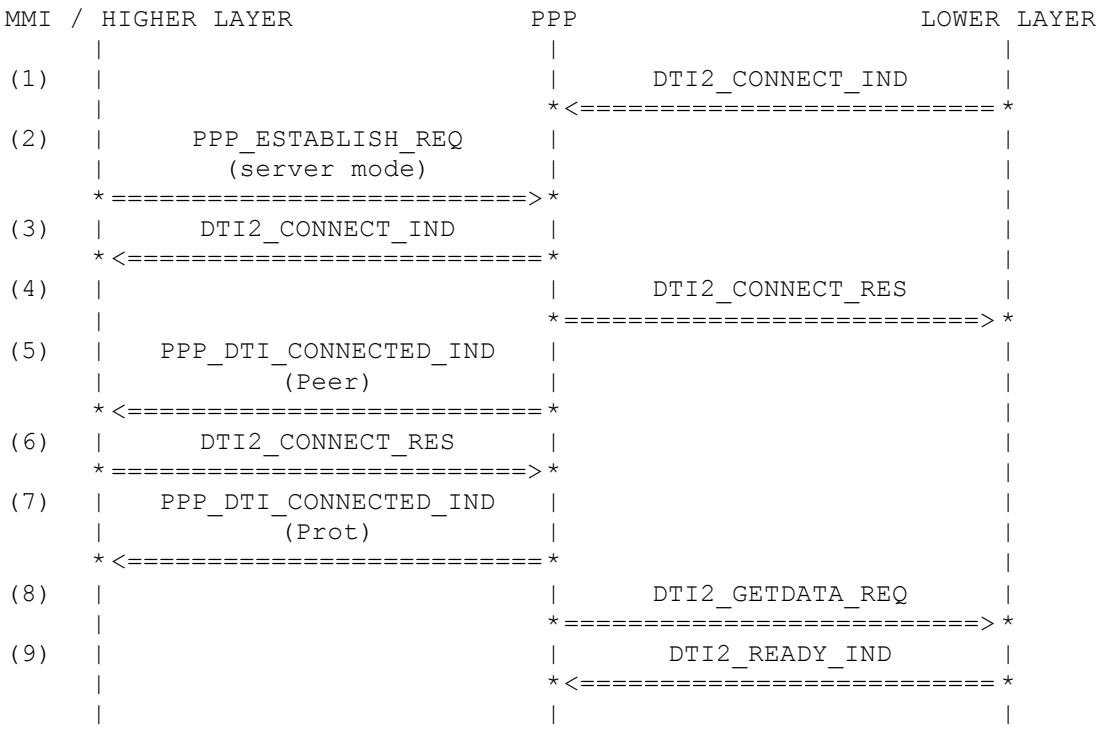
<A>Establishment in server mode whith automatic authentication protocol negotiation.
<B>Establishment in server mode whith CHAP authentication.
<C>Establishment in server mode whith PAP authentication.
<D>Establishment in server mode whithout any authentication.
<E>Establishment in server mode whith automatic authentication and Async-Control-Character-Map = 0.
<F>Establishment in server mode whith MRU = 2000 and ACCM = 0.

Variants:

<A>….<F>

Preamble:

PPP001



Parametrization:

Primitive	Parameter	Value
(1) DTI2_CONNECT_IND		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(2) PPP_ESTABLISH_REQ		
	mode	PPP_SERVER
<A>	mru	PPP_MRU_DEFAULT
<B>	mru	PPP_MRU_DEFAULT
<C>	mru	PPP_MRU_DEFAULT
<D>	mru	PPP_MRU_DEFAULT
<E>	mru	PPP_MRU_DEFAULT
<F>	mru	PPP_MRU_2000
<A>	ap	PPP_AP_AUTO
<B>	ap	PPP_AP_CHAP
<C>	ap	PPP_AP_PAP
<D>	ap	PPP_AP_NO
<E>	ap	PPP_AP_AUTO
<F>	ap	PPP_AP_AUTO
	login	AUTH_STRUCT0
<A>	accm	PPP_ACCM_DEFAULT
<B>	accm	PPP_ACCM_DEFAULT
<C>	accm	PPP_ACCM_DEFAULT
<D>	accm	PPP_ACCM_DEFAULT
<E>	accm	PPP_ACCM_0
<F>	accm	PPP_ACCM_0
	rt	PPP_RT_DEFAULT
	mc	PPP_MC_DEFAULT
	mt	PPP_MT_DEFAULT
	mf	PPP_MF_DEFAULT
	ppp_hc	PPP_HC_OFF
	ip	PPP_IP_DYNAMIC
	dns1	PPP_DNS1_DYNAMIC

	dns2	PPP_DNS2_DYNAMIC
	peer_channel	PEER_CHANNEL0
	protocol_channel	PROTOCOL_CHANNEL0
	peer_direction	
	DTI_CHANNEL_TO_LOWER_LAYER	
	prot_direction	
	DTI_CHANNEL_TO_HIGHER_LAYER	
	peer_link_id	LINK_ID_PEER
	prot_link_id	LINK_ID_PROT
(3) DTI2_CONNECT_IND		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(4) DTI2_CONNECT_RES		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(5) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PEER
(6) DTI2_CONNECT_RES		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(7) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PROT
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(9) DTI2_READY_IND		
	link_id	LINK_ID_PEER

History:

10-Oct-2000	STW	Initial
22-Jun-2001	XOF	Channel ID and connect primitive added.
12-Jul-2001	STW	Remove NOT_USED values.
13-Sep-2001	XOF	Build DTILIB functionality
24-Oct-2002	STW	Add variant (MRU = 2000, ACCM = 0).

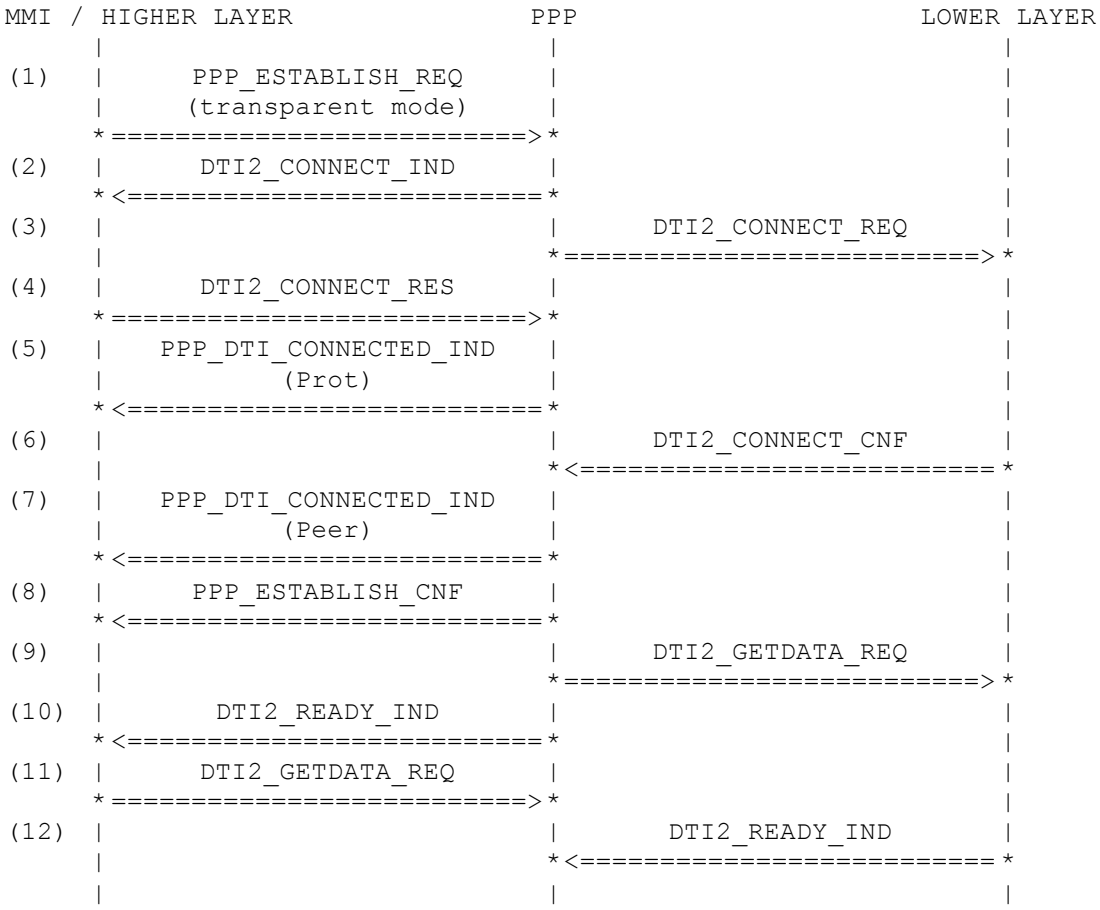
4.2.4 PPP014: Link establishment in transparent mode

Description:

Establishment in transparent mode.

Preamble:

PPP001



Parametrization:

Primitive	Parameter	Value
(1) PPP_ESTABLISH_REQ		
	mode	PPP_TRANSPARENT
	mru	PPP_MRU_DEFAULT
	ap	PPP_AP_NO
	login	AUTH_STRUCT0
	accm	PPP_ACCM_DEFAULT
	rt	PPP_RT_DEFAULT
	mc	PPP_MC_DEFAULT
	mt	PPP_MT_DEFAULT
	mf	PPP_MF_DEFAULT
	ppp_hc	PPP_HC_OFF
	ip	PPP_IP_DYNAMIC
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
	peer_channel	PEER_CHANNEL0
	protocol_channel	PROTOCOL_CHANNEL0
	peer_direction	DTI_CHANNEL_TO_LOWER_LAYER
	prot_direction	DTI_CHANNEL_TO_HIGHER_LAYER
	peer_link_id	LINK_ID_PEER
	prot_link_id	LINK_ID_PROT
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_PROT

		version	
	DTI_VERSION_10		
(3) DTI2_CONNECT_REQ			
		link_id	LINK_ID_PEER
		version	
	DTI_VERSION_10		
(4) DTI2_CONNECT_RES			
	link_id	LINK_ID_PROT	
	version	DTI_VERSION_10	
(5) PPP_DTI_CONNECTED_IND			
	connected_direction	PPP_DTI_CONN_PROT	
(6) DTI2_CONNECT_CNF			
	link_id	LINK_ID_PEER	
	version	DTI_VERSION_10	
(7) PPP_DTI_CONNECTED_IND			
	connected_direction	PPP_DTI_CONN_PEER	
(8) PPP_ESTABLISH_CNF			
	mru	PPP_MRU_DEFAULT	
	ppp_hc	PPP_HC_OFF	
	msid	MSID_DUMMY	
	ip	PPP_IP_DYNAMIC	
	dns1	PPP_DNS1_DYNAMIC	
	dns2	PPP_DNS2_DYNAMIC	
(9) DTI2_GETDATA_REQ			
	link_id	LINK_ID_PEER	
(10) DTI2_READY_IND			
	link_id	LINK_ID_PROT	
(11) DTI2_GETDATA_REQ			
	link_id	LINK_ID_PROT	
(12) DTI2_READY_IND			
	link_id	LINK_ID_PEER	

History:

20-Oct-2000	STW	Initial
22-Jun-2001	XOF	Channel ID and connect primitive added.
12-Jul-2001	STW	Remove NOT_USED values.
13-Sept-2001	XOF	Build DTILIB functionality

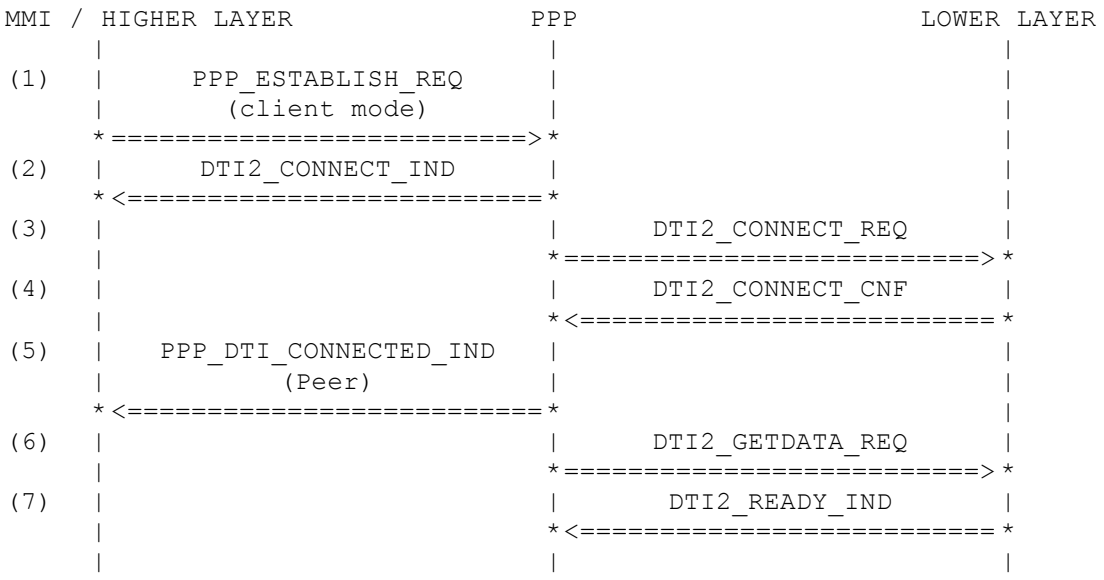
4.2.5 PPP015: Link establishment in client mode (PAP), only peer dti channel opened

Description:

Establishment in client mode whithout any authentification.

Preamble:

PPP001



Parametrization:

Primitive	Parameter	Value
(1) PPP_ESTABLISH_REQ		
	mode	PPP_CLIENT
	mru	PPP_MRU_DEFAULT
	ap	PPP_AP_PAP
	login	AUTH_STRUCT0
	accm	PPP_ACCM_DEFAULT
	rt	PPP_RT_DEFAULT
	mc	PPP_MC_DEFAULT
	mt	PPP_MT_DEFAULT
	mf	PPP_MF_DEFAULT
	ppp_hc	PPP_HC_OFF
	ip	PPP_IP_DYNAMIC
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
	peer_channel	PEER_CHANNEL0
	protocol_channel	PROTOCOL_CHANNEL0
	peer_direction	DTI_CHANNEL_TO_LOWER_LAYER
	prot_direction	DTI_CHANNEL_TO_HIGHER_LAYER
	peer_link_id	LINK_ID_PEER
	prot_link_id	LINK_ID_PROT
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(3) DTI2_CONNECT_REQ		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(4) DTI2_CONNECT_CNF		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(5) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PEER

(6)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(7)	DTI2_READY_IND		
		link_id	LINK_ID_PEER

History:

25-Apr-2002	TVO	Initial

4.3 LCP Establishment (PPP020-PPP029)

4.3.1 PPP020: LCP Establishment (Client, PAP)

Description:

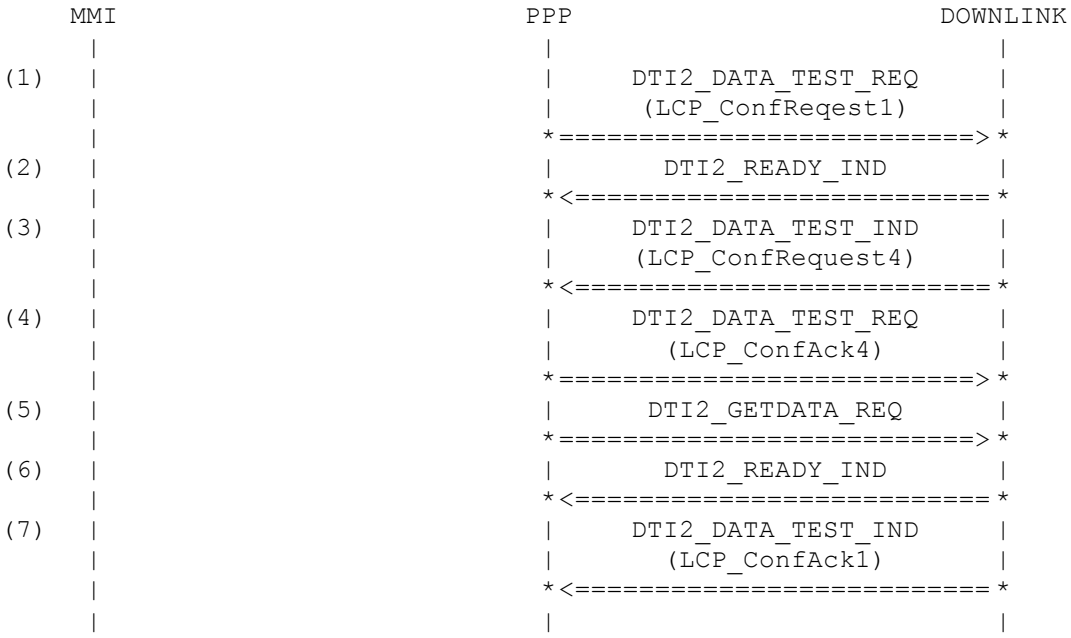
Peer configures PAP.

Variants:

<A>....<B>

Preamble:

<A>PPP012
<B>PPP015



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest1
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER

		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfRequest4
(4)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAck4
(5)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(6)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(7)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAck1

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

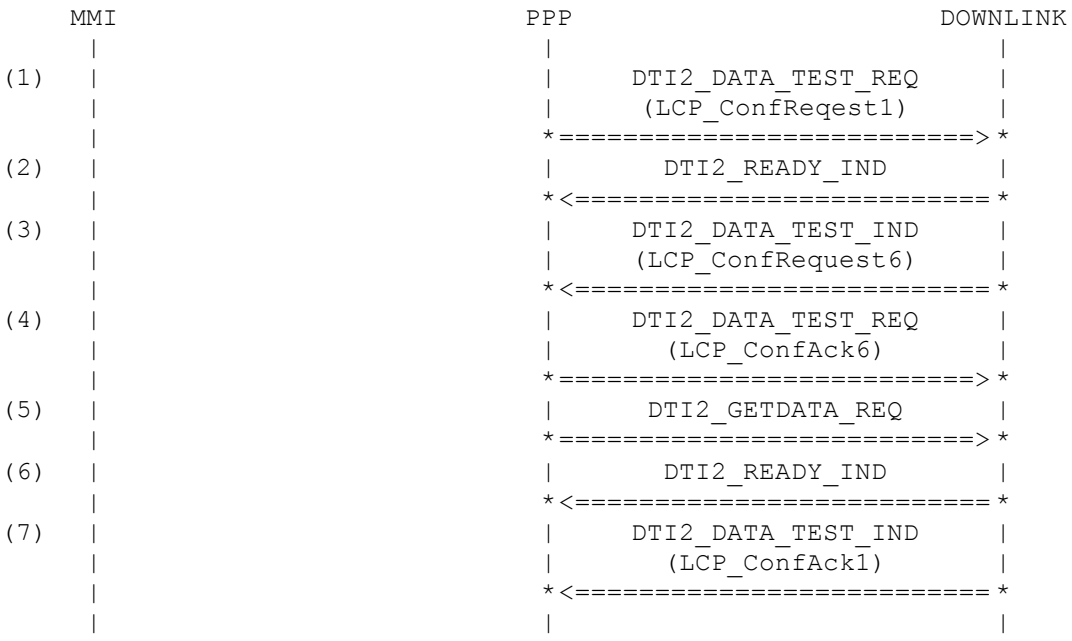
4.3.2 PPP021: LCP Establishment (CLient, PAP, ACFC, PFC)

Description:

Peer configures PAP,ACFC (Address-and-Control-Field-Compression) am PFC (Protocol-Field-Compression).

Preamble:

PPP012



Parametrization:

Primitive	Parameter	Value

(1)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfRequest1
(2)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(3)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfRequest6
(4)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAck6
(5)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(6)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(7)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAck1

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

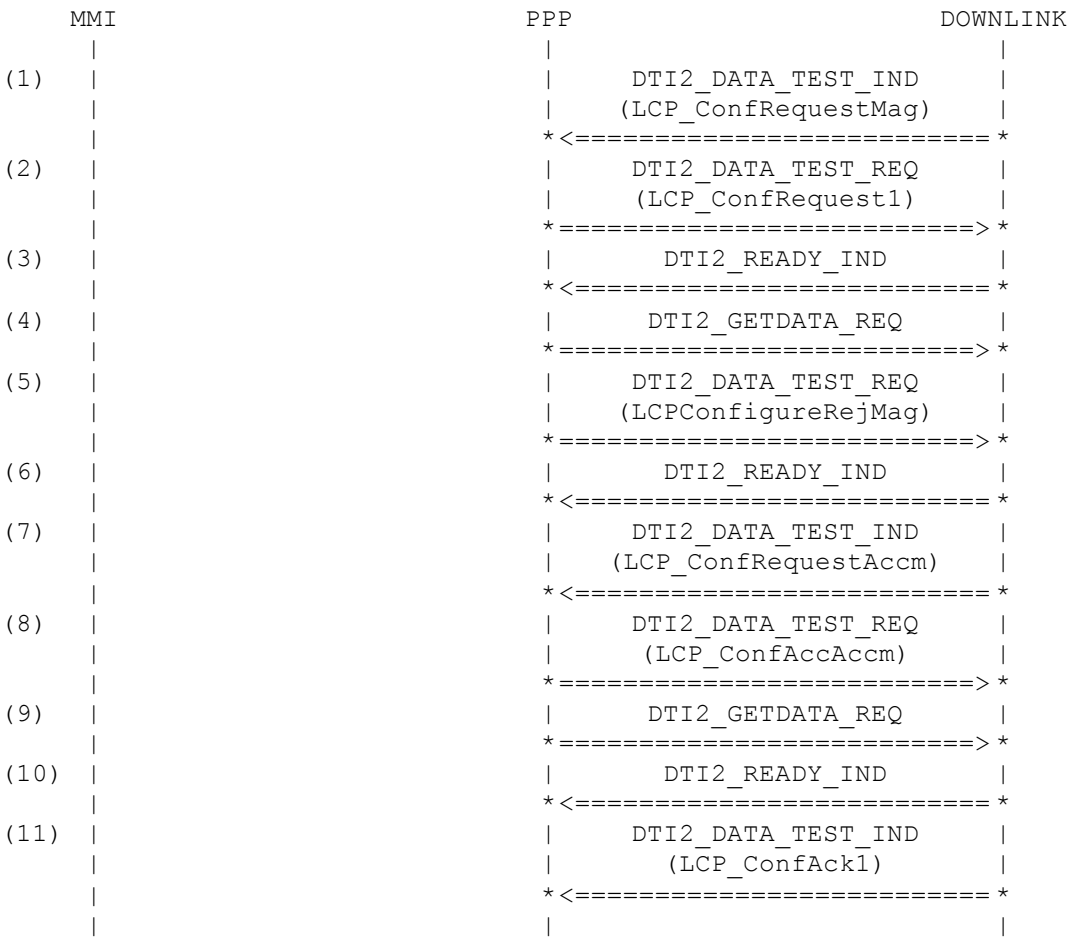
4.3.3 PPP022: LCP Establishment (CLient, PAP, ACFC, PFC) with magic number request

Description:

Peer configures PAP,ACFC (Address-and-Control-Field-Compression) am PFC (Protocol-Field-Compression). Magic number is rejected by PPP client.

Preamble:

PPP012



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequestMag
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest1
(3) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCPConfigureRejMag
(5) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER

	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequestAccm
(7) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfAccAccm
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(9) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfAck1

History:

26-Sep-2000	MT	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.3.4 PPP023: LCP Establishment (Server, CHAP, ACFC, PFC) with magic number request

Description:

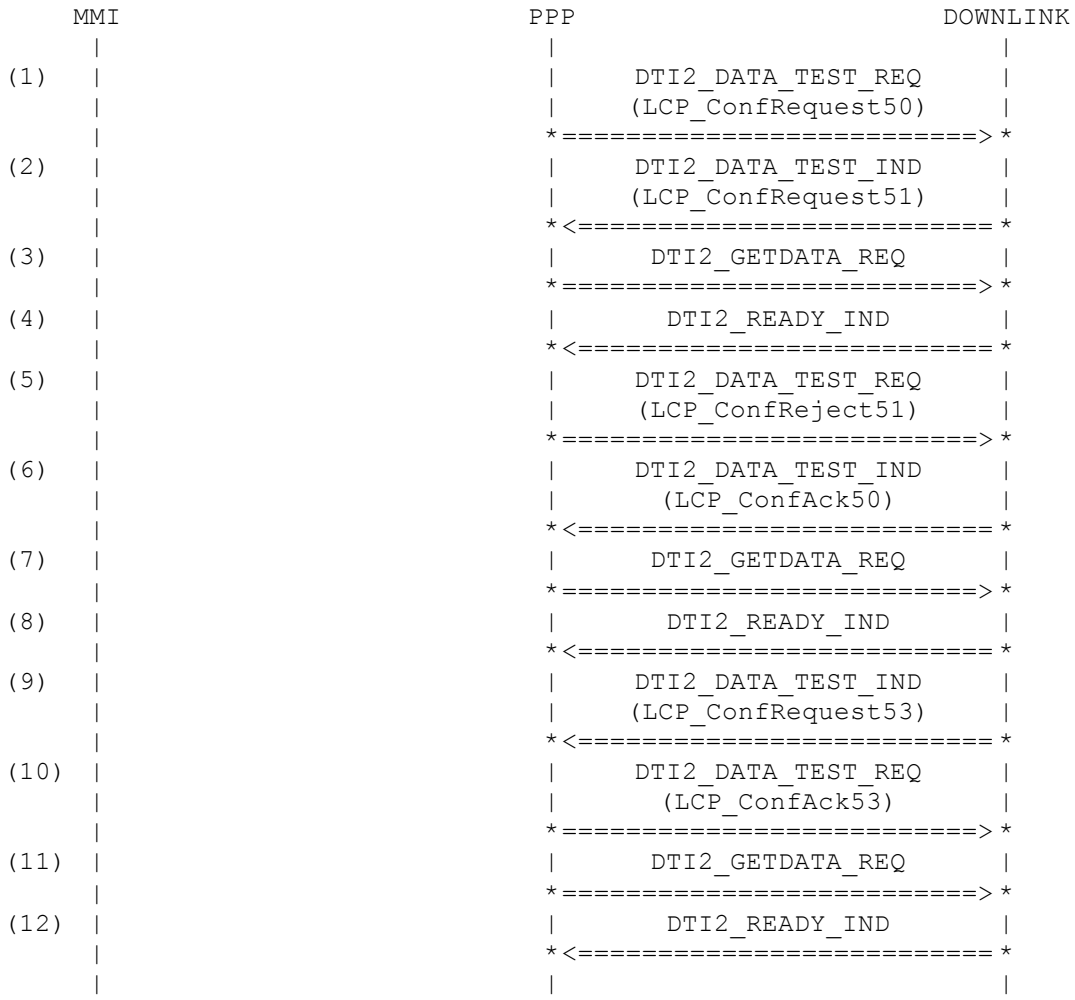
Server configures
<A>CHAP
<B>CHAP
<C>PAP
<D>no authentication
<E>CHAP and ACCM=0
and ACFC (Address-and-Control-Field-Compression) am PFC (Protocol-Field-Compression). Magic number is rejected by PPP server.

Variants:

<A>….<E>

Preamble:

<A>PPP013A
<B>PPP013B
<C>PPP013C
<D>PPP013D
<E>PPP013E



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
	<A>	sdu	LCP_ConfRequest50
	<B>	sdu	LCP_ConfRequest50
	<C>	sdu	LCP_ConfRequest66
	<D>	sdu	LCP_ConfRequest68
	<E>	sdu	LCP_ConfRequest74
(2)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfRequest51
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(4)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(5)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfReject51

(6)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
	<A>	sdu	LCP_ConfAck50
	<B>	sdu	LCP_ConfAck50
	<C>	sdu	LCP_ConfAck66
	<D>	sdu	LCP_ConfAck68
	<E>	sdu	LCP_ConfAck74
(7)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(8)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(9)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfRequest53
(10)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAck53
(11)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(12)	DTI2_READY_IND		
		link_id	LINK_ID_PEER

History:

10-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.3.5 PPP024: LCP Establishment (Server) re-request of rejected configuration options

Description:

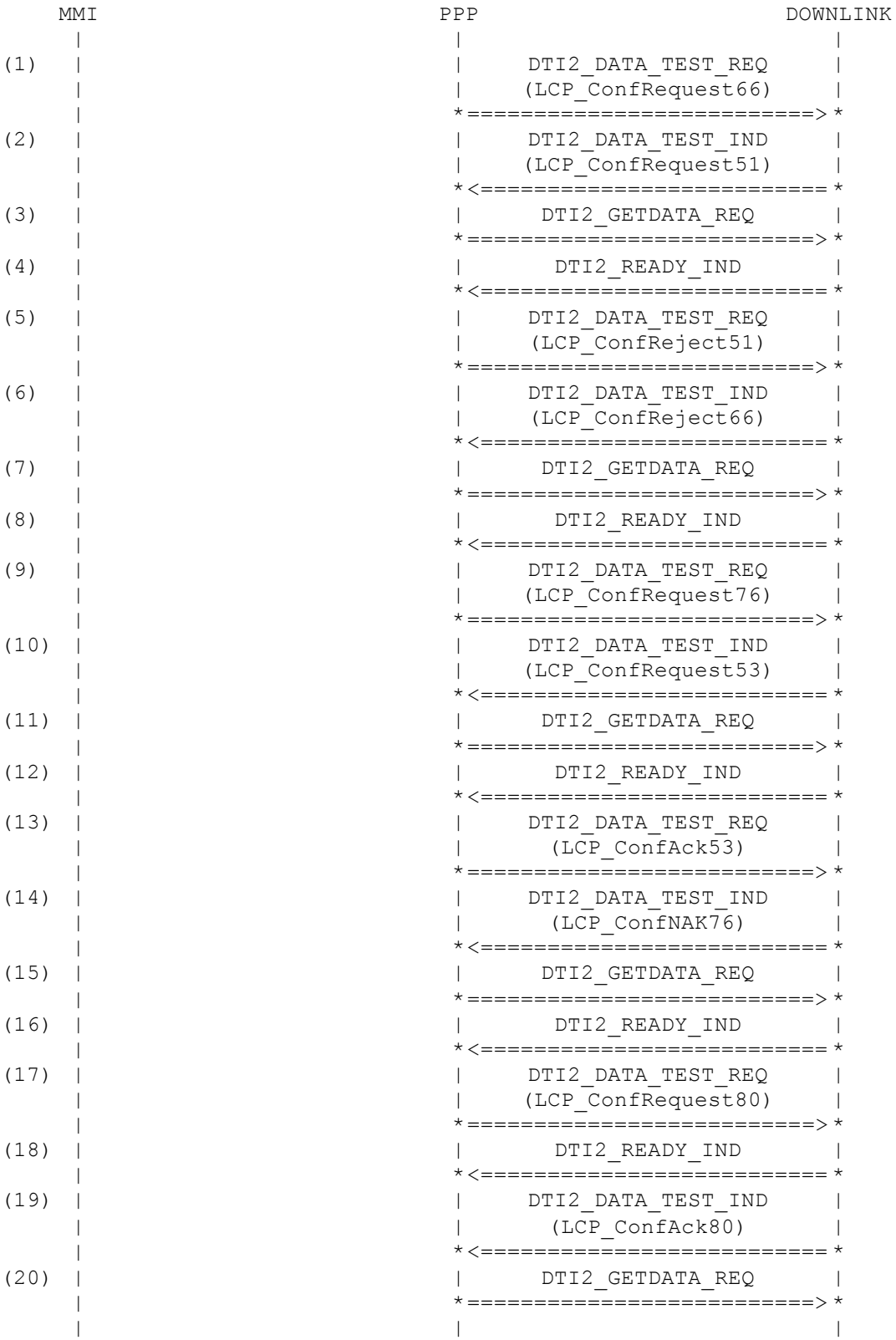
Re-request of already rejected configuration options
<A>PFC
<B>ACFC
<C>MRU
<D>ACCM
<E>Authentication Protocol

Variants:

<A>….<E>

Preamble:

<A>PPP013C
<B>PPP013C
<C>PPP013F
<D>PPP013F
<E>PPP013F



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER

<A>	sdu	LCP_ConfRequest66
<B>	sdu	LCP_ConfRequest66
<C>	sdu	LCP_ConfRequest82
<D>	sdu	LCP_ConfRequest82
<E>	sdu	LCP_ConfRequest82
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest51
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfReject51
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ConfReject66A
<B>	sdu	LCP_ConfReject66B
<C>	sdu	LCP_ConfReject82A
<D>	sdu	LCP_ConfReject82B
<E>	sdu	LCP_ConfReject82C
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ConfRequest76
<B>	sdu	LCP_ConfRequest78
<C>	sdu	LCP_ConfRequest84
<D>	sdu	LCP_ConfRequest86
<E>	sdu	LCP_ConfRequest90
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest53
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(13) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfAck53
(14) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ConfNAK76

<B>	sdu	LCP_ConfNAK78
<C>	sdu	LCP_ConfNAK84
<D>	sdu	LCP_ConfNAK86
<E>	sdu	LCP_ConfNAK90
(15) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(16) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(17) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ConfRequest80
<B>	sdu	LCP_ConfRequest80
<C>	sdu	LCP_ConfRequest88
<D>	sdu	LCP_ConfRequest88
<E>	sdu	LCP_ConfRequest88
(18) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(19) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ConfAck80
<B>	sdu	LCP_ConfAck80
<C>	sdu	LCP_ConfAck88
<D>	sdu	LCP_ConfAck88
<E>	sdu	LCP_ConfAck88
(20) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

15-Aug-2001	STW	Initial
24-Oct-2002	STW	Adaptation to DTI2

4.4 Authentication Procedure (PPP030-PPP039)

4.4.1 PPP030: PAP Authentication in Client Mode

Description:

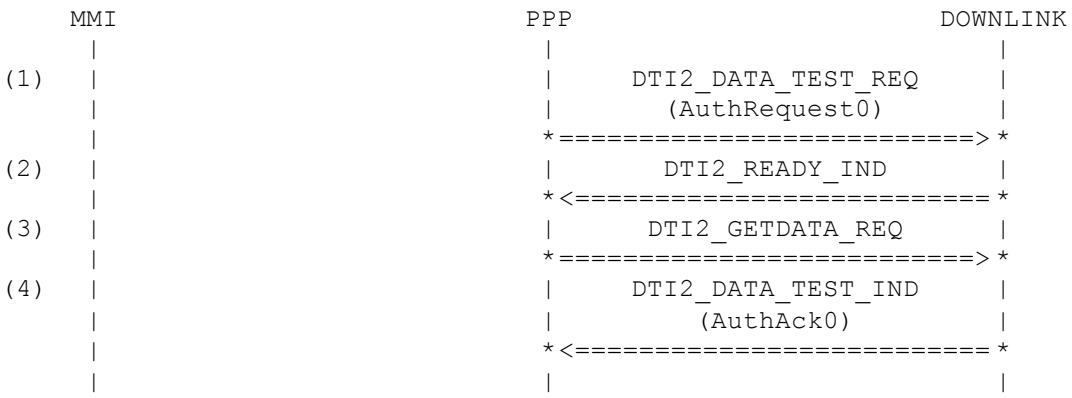
PAP authentication in client mode without Address-and-Control-Field-Compression and Protocol-Field-Compression

Variants:

<A>...<B>

Preamble:

<A>PPP020A
<B>PPP020B



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthRequest0
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthAck0

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

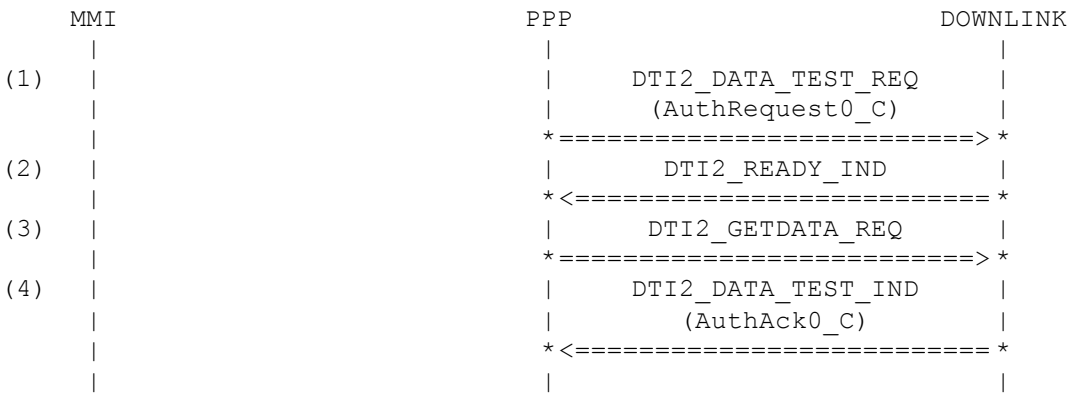
4.4.2 PPP031: PAP Authentication in Client Mode (ACFC, PFC)

Description:

PAP authentication in client mode with Address-and-Control-Field-Compression and Protocol-Field-Compression

Preamble:

PPP021



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthRequest0_C
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthAck0_C

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

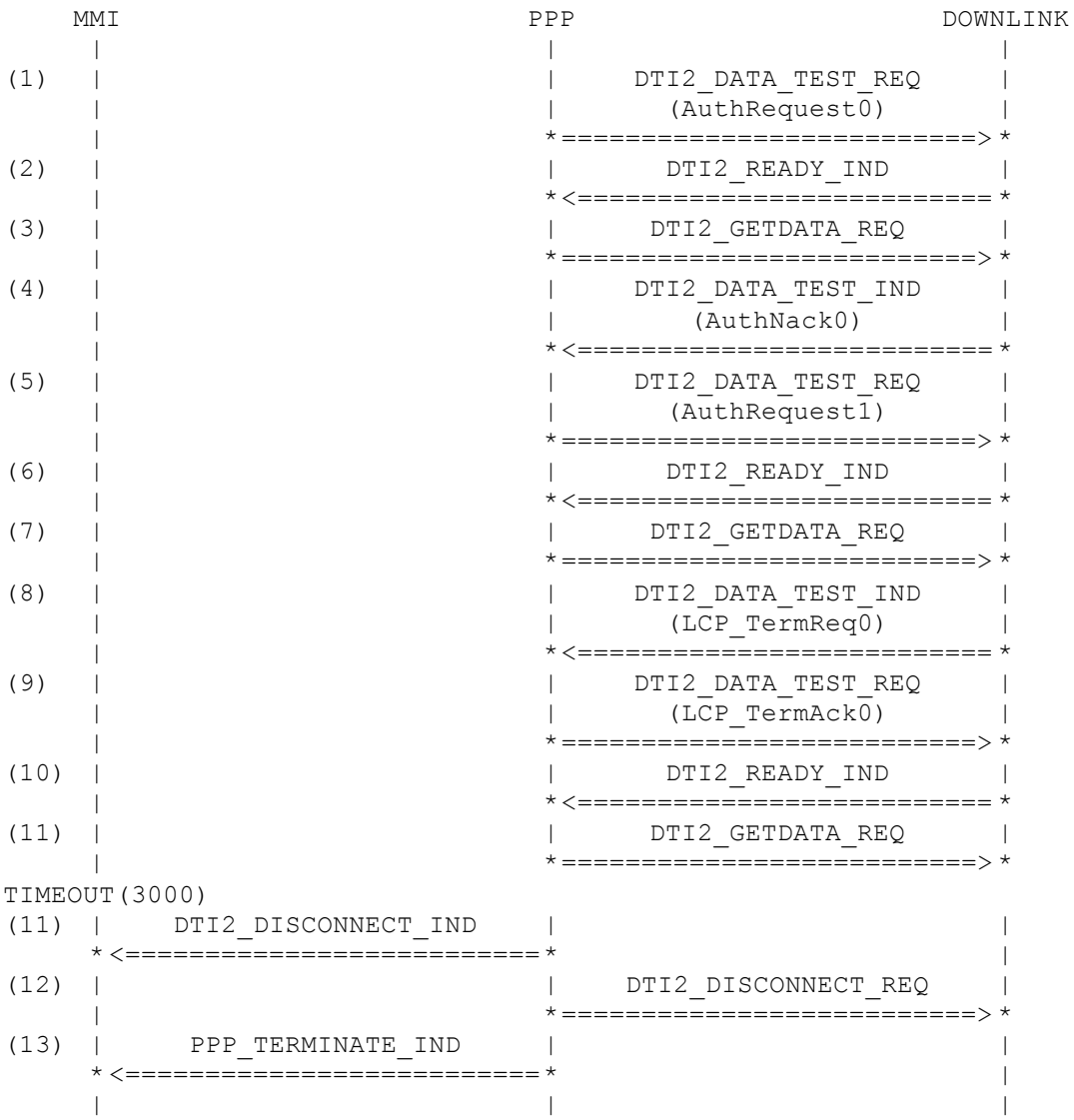
4.4.3 PPP032: PAP authtentication failed (client mode)

Description:

PAP authentication in client mode, but the server NAK's authentication and terminates the link.

Preamble:

PPP020A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthRequest0
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthNack0
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	AuthRequest1

(6) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermReq0
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermAck0
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_PROT
	cause	
	DTI_CAUSE_NORMAL_CLOSE	
(13) DTI2_DISCONNECT_REQ		
	link_id	LINK_ID_PEER
	cause	
	DTI_CAUSE_NORMAL_CLOSE	
(14) PPP_TERMINATE_IND		
	ppp_cause	
	PPP_TERM_USE_AUTHED_FAILED	

History:

22-Aug-2000	SG	Initial
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality
24-Oct-2002	STW	Cause concept

4.4.4 PPP033: CHAP Authentication in Server Mode (ACFC, PFC) with LCP Identification

Description:

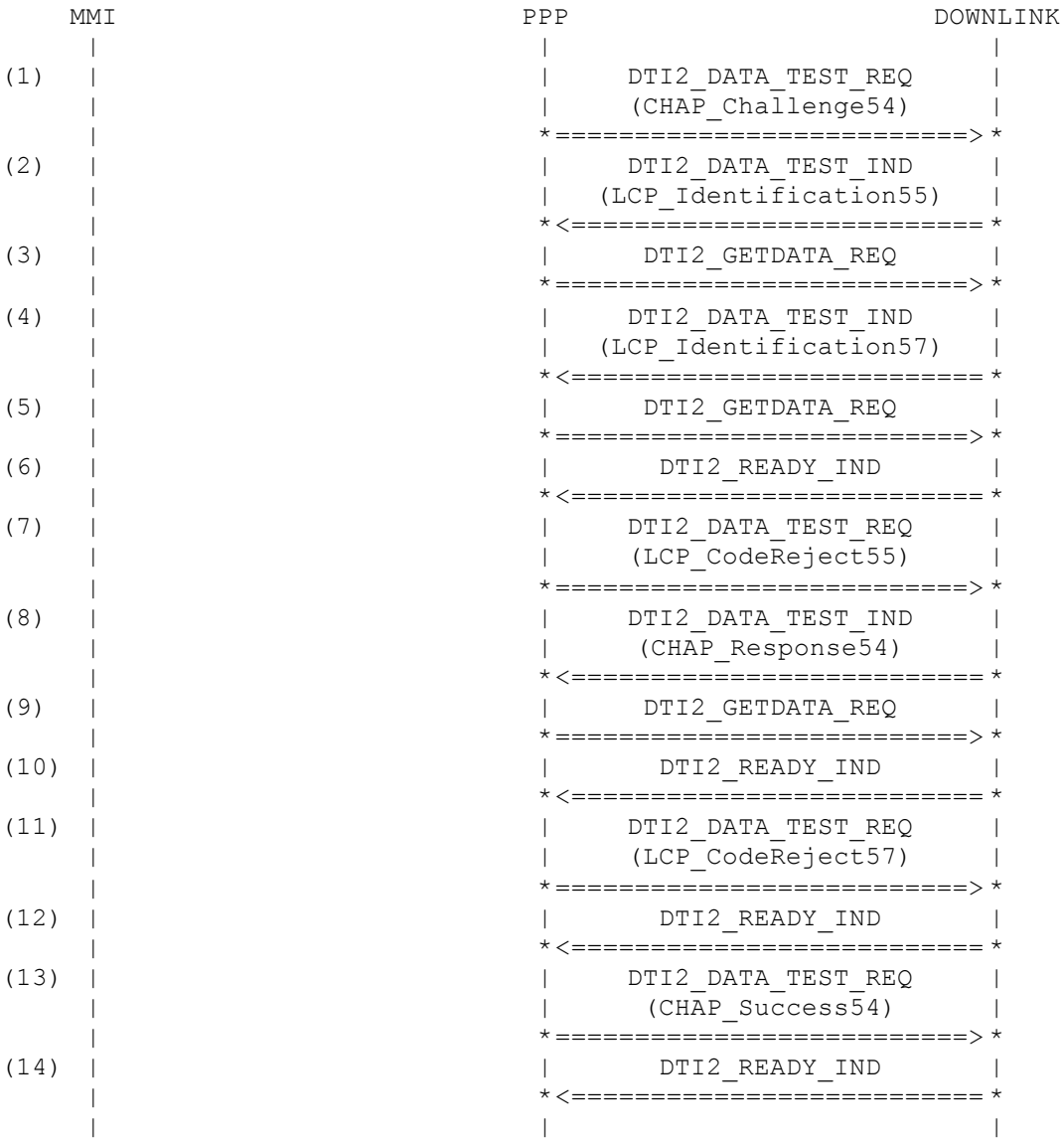
CHAP authentication in server mode. The client sends LCP Identification packets which will be rejected.
<A>with ACFC, PFC
<B>with ACFC, PFC
<C>with ACFC, PFC and ACCM=0

Variants:

<A>....<C>

Preamble:

<A>PPP023A
<B>PPP023B
<C>PPP023E



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	CHAP_Challenge54
<B>	sdu	CHAP_Challenge54
<C>	sdu	CHAP_Challenge76
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_Identification55
<B>	sdu	LCP_Identification55
<C>	sdu	LCP_Identification77
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER

		parameters	DTI_PARAMETER_PEER
	<A>	sdu	LCP_Identification57
	<B>	sdu	LCP_Identification57
	<C>	sdu	LCP_Identification79
(5)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(6)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(7)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_CodeReject55
(8)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
	<A>	sdu	CHAP_Response54
	<B>	sdu	CHAP_Response54
	<C>	sdu	CHAP_Response76
(9)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(10)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(11)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_CodeReject57
(12)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(13)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
	<A>	sdu	CHAP_Success54
	<B>	sdu	CHAP_Success54
	<C>	sdu	CHAP_Success76
(14)	DTI2_READY_IND		
		link_id	LINK_ID_PEER

History:

10-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

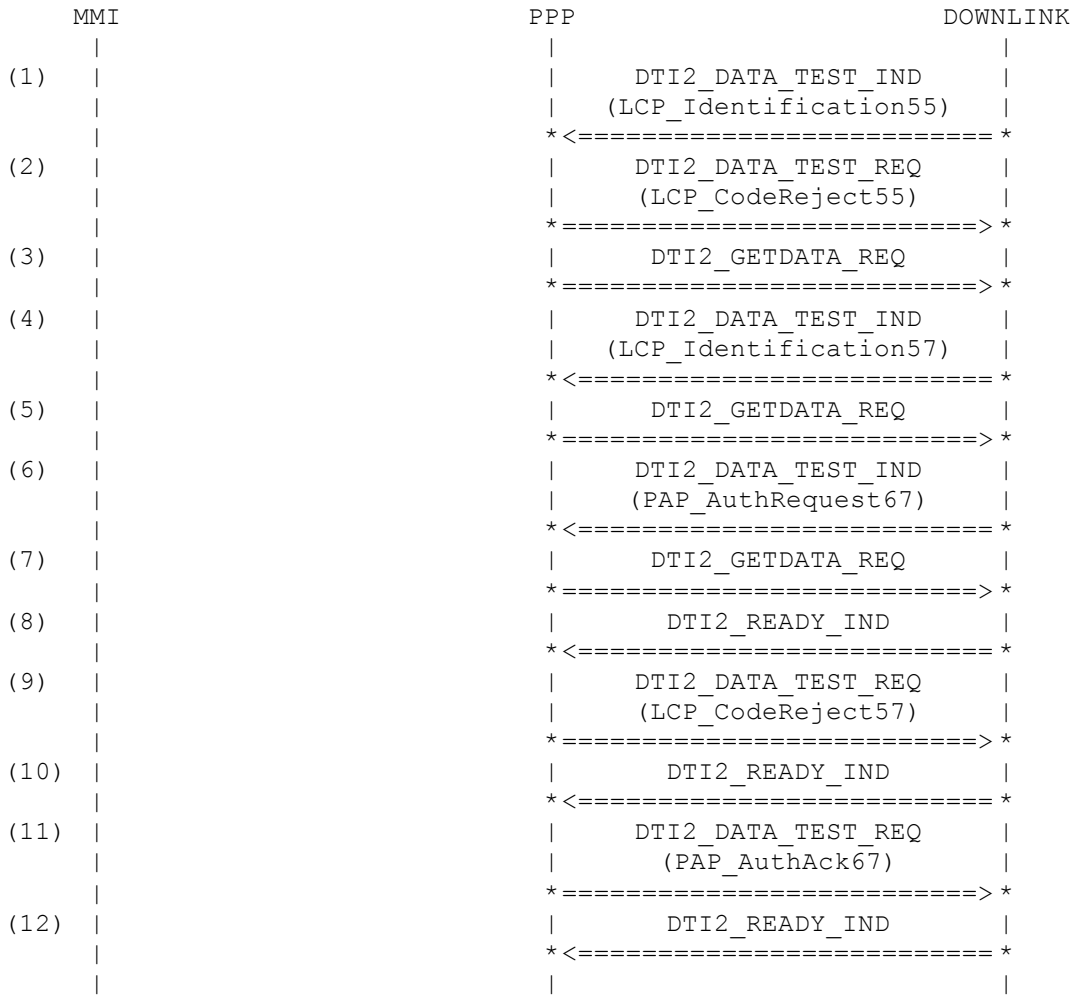
4.4.5 PPP034: PAP Authentication in Server Mode (ACFC, PFC) with LCP Identification

Description:

PAP authentication in server mode. The client sends LCP Identification packets which will be rejected.

Preamble:

PPP023C



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_Identification55
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_CodeReject55
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_Identification57
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	PAP_AuthRequest67

(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_CodeReject57
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	PAP_AuthAck67
(12) DTI2_READY_IND		
	link_id	LINK_ID_PEER

History:

13-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.5 IPCP Establishment (PPP40-PPP049)

4.5.1 PPP040: Normal IPCP Address negotiation

Description:

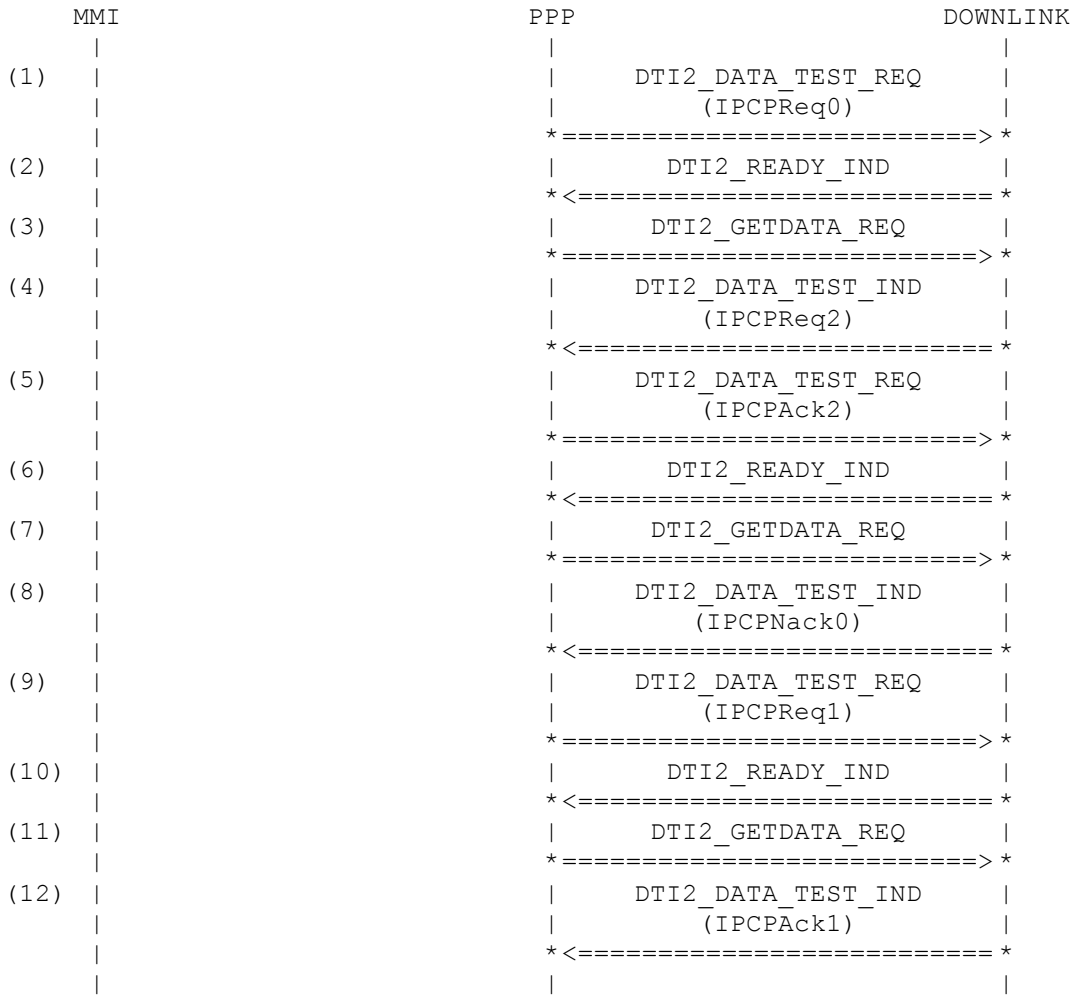
Successful IPCP negotiation procedure.

Variants:

<A>….<B>

Preamble:

<A>PPP030A
<B>PPP030B



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq0
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq2
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck2
(6) DTI2_READY_IND		
	link_id	LINK_ID_PEER

(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPNack0
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq1
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck1

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

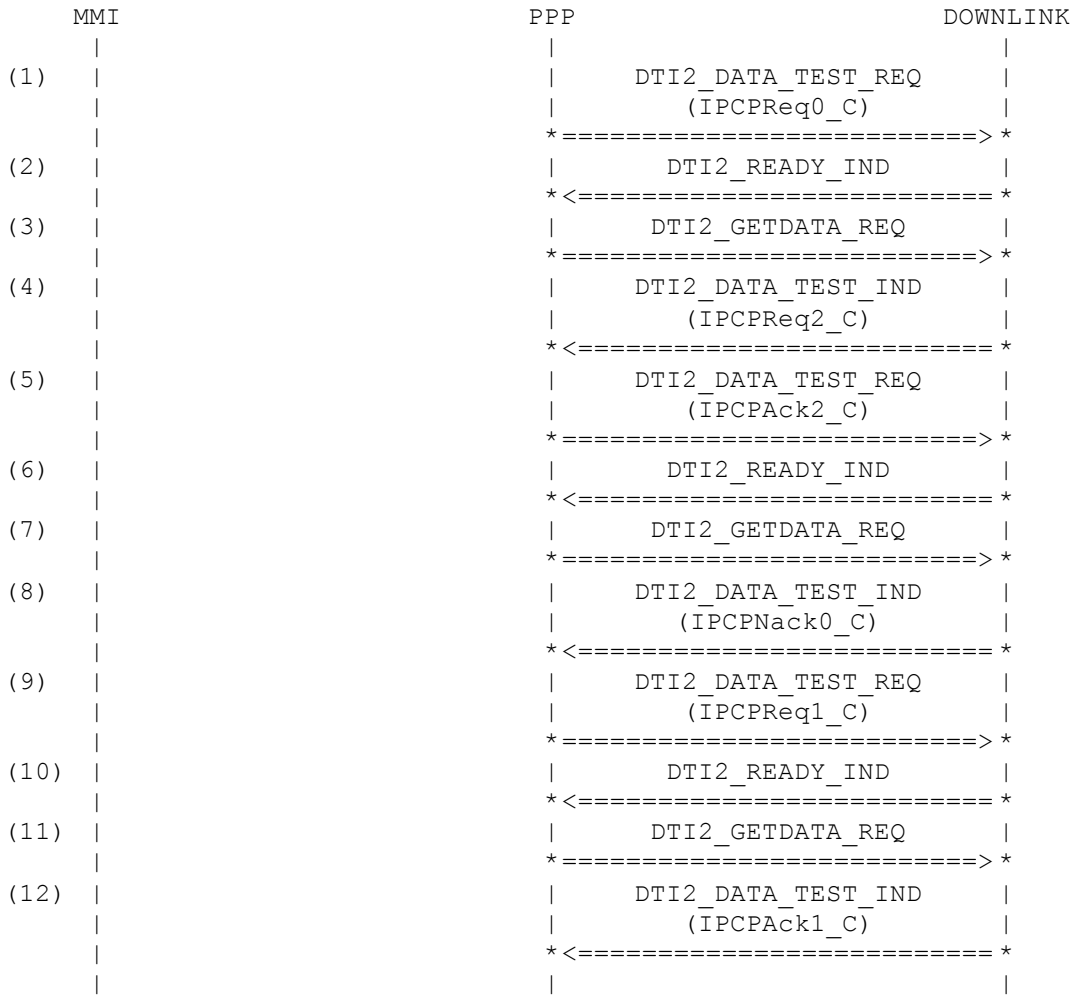
4.5.2 PPP041: Normal IPCP Address negotiation (ACFC, PFC)

Description:

Successful IPCP negotiation procedure, ACFC and PFC enabled.

Preamble:

PPP031



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq0_C
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq2_C
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck2_C
(6) DTI2_READY_IND		
	link_id	LINK_ID_PEER

(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPNack0_C
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq1_C
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck1_C

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

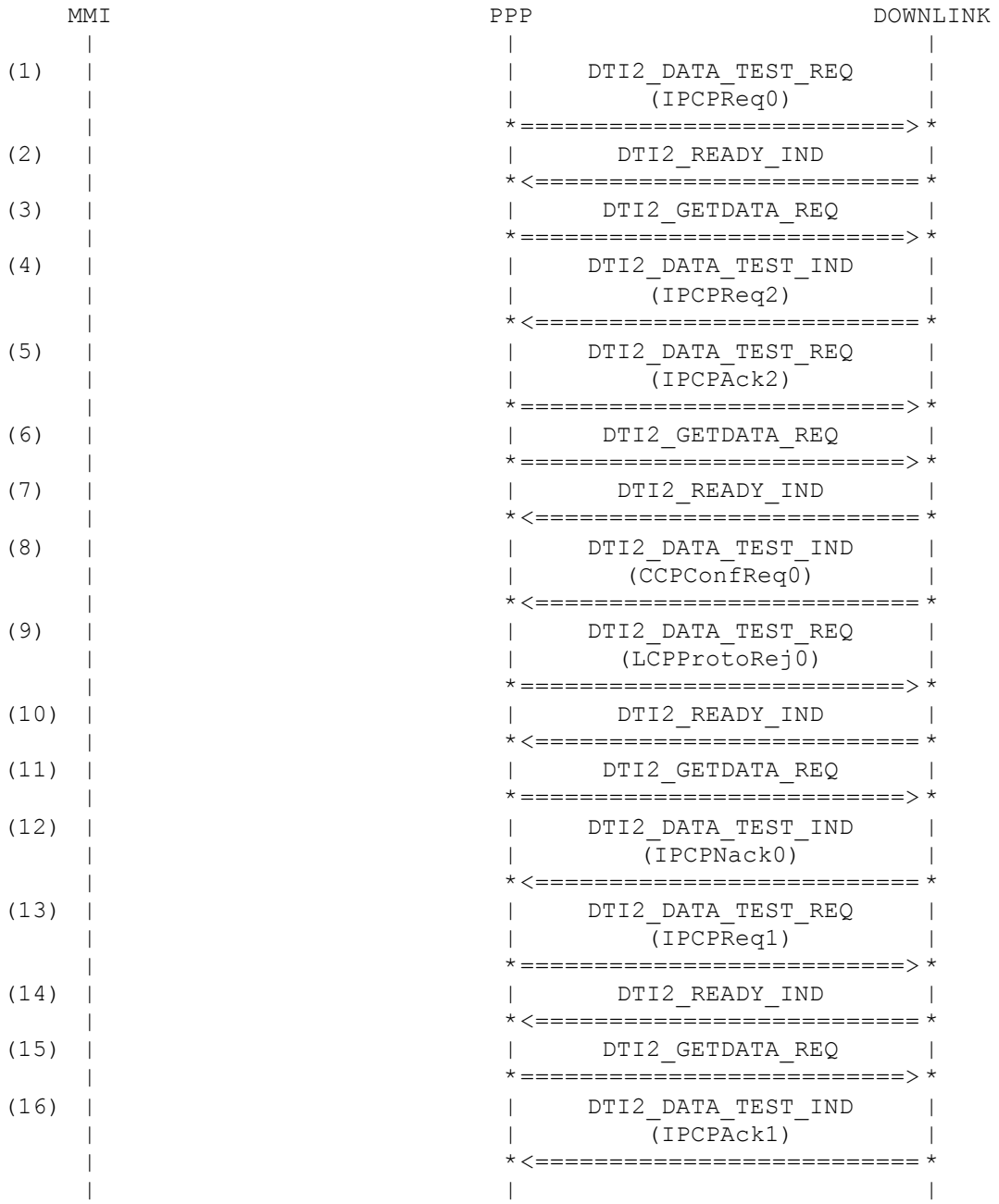
4.5.3 PPP042: CCP request from server during ICPC procedure

Description:

PPP receives a CCP (Compression Control Protocol) configure request and sends a LCP protocol reject. The IPCP negotiation procedure continues.

Preamble:

PPP030A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq0
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER

	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq2
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck2
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(7) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	CCPConfReq0
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCPProtoRej0
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPNack0
(13) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq1
(14) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(15) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(16) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck1

History:

22-Aug-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.5.4 PPP043: IPCP configuration in server mode with CCP

Description:

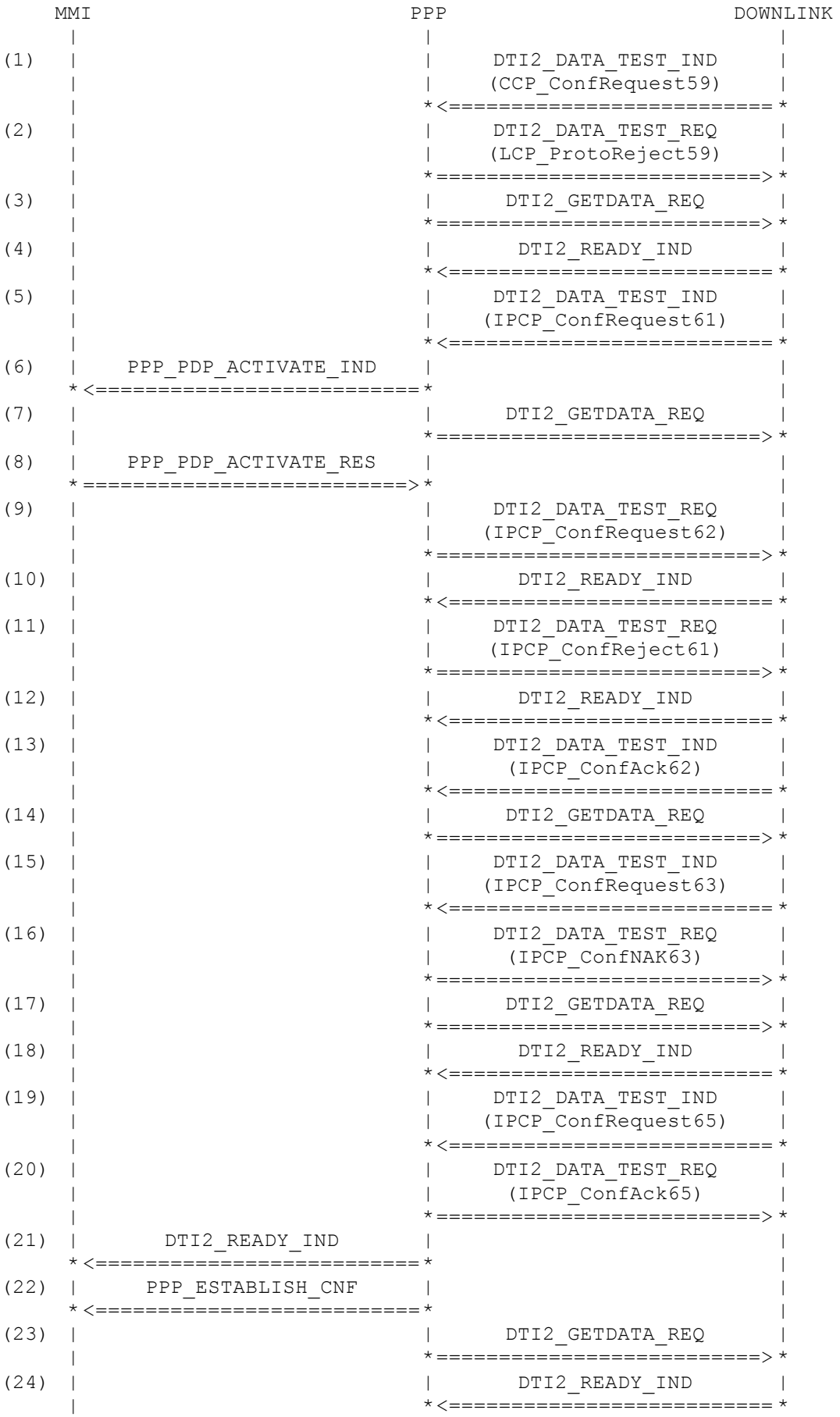
IPCP configuration in server mode. PPP receives a CCP (Compression Control Protocol) configure request and sends a LCP protocol reject. The IPCP negotiation procedure continues.
<A>automatic CHAP
<B>CHAP
<C>PAP with PCO sesponse with offset
<D>whithout authentication
<E>without Header Compression – Peer initiated
<F>without Header Compression – MMI initiated
<G>CHAP with ACCM=0

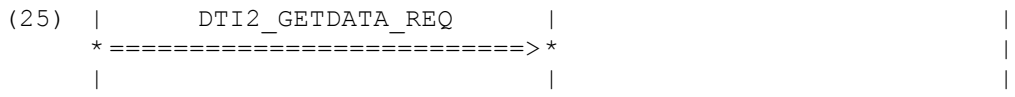
Variants:

<A>....<G>

Preamble:

<A>PPP033A
<B>PPP033B
<C>PPP034
<D>PPP023D
<E>PPP033A
<F>PPP033B
<G>PPP033C


MMI
PPP
DOWNLINK
(1)
DTI2_DATA_TEST_IND
(CCP_ConfRequest59)
(2)
DTI2_DATA_TEST_REQ
(LCP_ProtoReject59)
(3)
DTI2_GETDATA_REQ
(4)
DTI2_READY_IND
(5)
DTI2_DATA_TEST_IND
(IPCP_ConfRequest61)
(6)
PPP_PDP_ACTIVATE_IND
(7)
DTI2_GETDATA_REQ
(8)
PPP_PDP_ACTIVATE_RES
(9)
DTI2_DATA_TEST_REQ
(IPCP_ConfRequest62)
(10)
DTI2_READY_IND
(11)
DTI2_DATA_TEST_REQ
(IPCP_ConfReject61)
(12)
DTI2_READY_IND
(13)
DTI2_DATA_TEST_IND
(IPCP_ConfAck62)
(14)
DTI2_GETDATA_REQ
(15)
DTI2_DATA_TEST_IND
(IPCP_ConfRequest63)
(16)
DTI2_DATA_TEST_REQ
(IPCP_ConfNAK63)
(17)
DTI2_GETDATA_REQ
(18)
DTI2_READY_IND
(19)
DTI2_DATA_TEST_IND
(IPCP_ConfRequest65)
(20)
DTI2_DATA_TEST_REQ
(IPCP_ConfAck65)
(21)
DTI2_READY_IND
(22)
PPP_ESTABLISH_CNF
(23)
DTI2_GETDATA_REQ
(24)
DTI2_READY_IND



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
	<A>	sdu	CCP_ConfRequest59
	<B>	sdu	CCP_ConfRequest59
	<C>	sdu	CCP_ConfRequest59
	<D>	sdu	CCP_ConfRequest59
	<E>	sdu	CCP_ConfRequest59
	<F>	sdu	CCP_ConfRequest59
	<G>	sdu	CCP_ConfRequest81
(2)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ProtoReject59
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(4)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(5)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
	<A>	sdu	IPCP_ConfRequest61
	<B>	sdu	IPCP_ConfRequest61
	<C>	sdu	IPCP_ConfRequest61
	<D>	sdu	IPCP_ConfRequest61
	<E>	sdu	IPCP_ConfRequest69
	<F>	sdu	IPCP_ConfRequest61
	<G>	sdu	IPCP_ConfRequest83
(6)	PPP_PDP_ACTIVATE_IND		
	<A>	ppp_hc	PPP_HC_VJ
	<B>	ppp_hc	PPP_HC_VJ
	<C>	ppp_hc	PPP_HC_VJ
	<D>	ppp_hc	PPP_HC_VJ
	<E>	ppp_hc	PPP_HC_OFF
	<F>	ppp_hc	PPP_HC_VJ
	<G>	ppp_hc	PPP_HC_VJ
	<A>	msid	MSID_EXPECTED
	<B>	msid	MSID_EXPECTED
	<C>	msid	MSID_EXPECTED
	<D>	msid	MSID_EXPECTED
	<E>	msid	MSID_DUMMY
	<F>	msid	MSID_EXPECTED
	<G>	msid	MSID_EXPECTED
	<A>	sdu	PCO_LIST_IND_CHAP
	<B>	sdu	PCO_LIST_IND_CHAP
	<C>	sdu	PCO_LIST_IND_PAP
	<D>	sdu	PCO_LIST_IND_NO
	<E>	sdu	PCO_LIST_IND_CHAP_NOVJ

<F>	sdu	PCO_LIST_IND_CHAP
<G>	sdu	PCO_LIST_IND_CHAP
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) PPP_PDP_ACTIVATE_RES		
<A>	ppp_hc	PPP_HC_VJ
<B>	ppp_hc	PPP_HC_VJ
<C>	ppp_hc	PPP_HC_VJ
<D>	ppp_hc	PPP_HC_VJ
<E>	ppp_hc	PPP_HC_OFF
<F>	ppp_hc	PPP_HC_OFF
<G>	ppp_hc	PPP_HC_VJ
<A>	msid	MSID_EXPECTED
<B>	msid	MSID_EXPECTED
<C>	msid	MSID_EXPECTED
<D>	msid	MSID_EXPECTED
<E>	msid	MSID_DUMMY
<F>	msid	MSID_DUMMY
<G>	msid	MSID_EXPECTED
	ip	PPP_IP_141_64_21_2
<A>	sdu	PCO_LIST_RES
<B>	sdu	PCO_LIST_RES
<C>	sdu	PCO_LIST_RES_OFFSET
<D>	sdu	PCO_LIST_RES
<E>	sdu	PCO_LIST_RES
<F>	sdu	PCO_LIST_RES
<G>	sdu	PCO_LIST_RES
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfRequest62
<B>	sdu	IPCP_ConfRequest62
<C>	sdu	IPCP_ConfRequest62
<D>	sdu	IPCP_ConfRequest62
<E>	sdu	IPCP_ConfRequest70
<F>	sdu	IPCP_ConfRequest70
<G>	sdu	IPCP_ConfRequest84
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfReject61
<B>	sdu	IPCP_ConfReject61
<C>	sdu	IPCP_ConfReject61
<D>	sdu	IPCP_ConfReject61
<E>	sdu	IPCP_ConfReject69
<F>	sdu	IPCP_ConfReject61F
<G>	sdu	IPCP_ConfReject83
(12) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(13) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfAck62
<B>	sdu	IPCP_ConfAck62

<C>	sdu	IPCP_ConfAck62
<D>	sdu	IPCP_ConfAck62
<E>	sdu	IPCP_ConfAck70
<F>	sdu	IPCP_ConfAck70
<G>	sdu	IPCP_ConfAck84
(14) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(15) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfRequest63
<B>	sdu	IPCP_ConfRequest63
<C>	sdu	IPCP_ConfRequest63
<D>	sdu	IPCP_ConfRequest63
<E>	sdu	IPCP_ConfRequest71
<F>	sdu	IPCP_ConfRequest71
<G>	sdu	IPCP_ConfRequest85
(16) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfNAK63
<B>	sdu	IPCP_ConfNAK63
<C>	sdu	IPCP_ConfNAK63
<D>	sdu	IPCP_ConfNAK63
<E>	sdu	IPCP_ConfNAK71
<F>	sdu	IPCP_ConfNAK71
<G>	sdu	IPCP_ConfNAK85
(17) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(18) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(19) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfRequest65
<B>	sdu	IPCP_ConfRequest65
<C>	sdu	IPCP_ConfRequest65
<D>	sdu	IPCP_ConfRequest65
<E>	sdu	IPCP_ConfRequest73
<F>	sdu	IPCP_ConfRequest73
<G>	sdu	IPCP_ConfRequest87
(20) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPCP_ConfAck65
<B>	sdu	IPCP_ConfAck65
<C>	sdu	IPCP_ConfAck65
<D>	sdu	IPCP_ConfAck65
<E>	sdu	IPCP_ConfAck73
<F>	sdu	IPCP_ConfAck73
<G>	sdu	IPCP_ConfAck87
(21) DTI2_READY_IND		
	link_id	LINK_ID_PROT
(22) PPP_ESTABLISH_CNF		
	mru	PPP_MRU_DEFAULT

<A>	ppp_hc	PPP_HC_VJ
<B>	ppp_hc	PPP_HC_VJ
<C>	ppp_hc	PPP_HC_VJ
<D>	ppp_hc	PPP_HC_VJ
<E>	ppp_hc	PPP_HC_OFF
<F>	ppp_hc	PPP_HC_OFF
<G>	ppp_hc	PPP_HC_VJ
<A>	msid	MSID_EXPECTED
<B>	msid	MSID_EXPECTED
<C>	msid	MSID_EXPECTED
<D>	msid	MSID_EXPECTED
<E>	msid	MSID_DUMMY
<F>	msid	MSID_DUMMY
<G>	msid	MSID_EXPECTED
	ip	PPP_IP_141_64_21_2
	dns1	PPP_PDNS_141_64_24_129
	dns2	PPP_SDNS_141_64_254_131
(23) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(24) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(25) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

12-Oct-2000	STW	Initial
22-Jun-2001	XOF	Channel ID added.
12-Jul-2001	STW	Add PCO with offset.
15-Sep-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

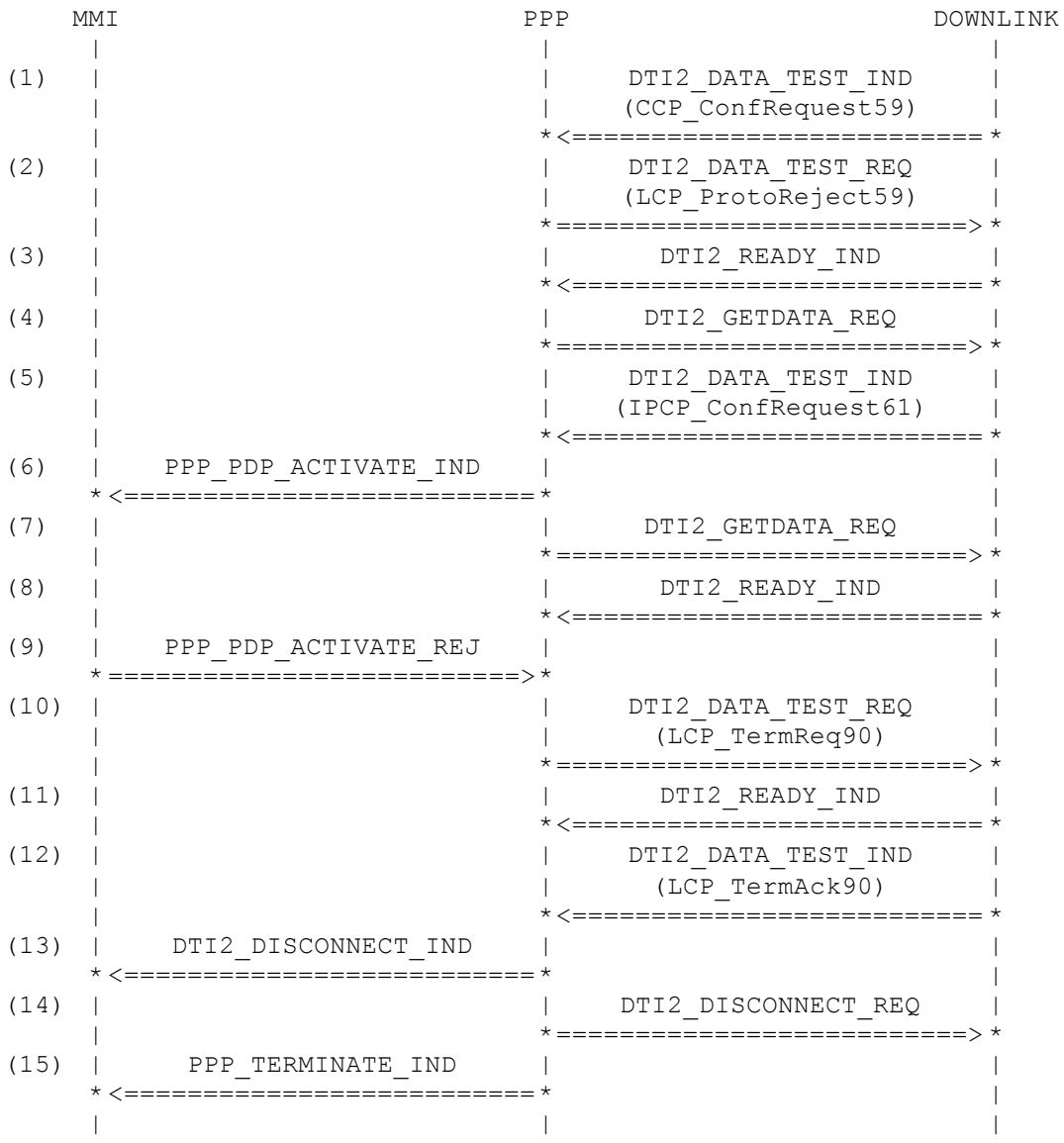
4.5.5 PPP044: PDP activation will be rejected by the network

Description:

IPCP configuration in server mode. PPP receives a CCP (Compression Control Protocol) configure request and sends a LCP protocol reject. The IPCP negotiation procedure continues, but PDP activation will be rejected by the network.

Preamble:

PPP033A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	CCP_ConfRequest59
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ProtoReject59
(3) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER

	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfRequest61
(6) PPP_PDP_ACTIVATE_IND		
	ppp_hc	PPP_HC_VJ
	msid	MSID_EXPECTED
	sdu	PCO_LIST_IND_CHAP
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(9) PPP_PDP_ACTIVATE_REJ		
	ppp_cause	PPP_TERM_INSUF_RES
(10) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermRequest90
(11) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermAck90
(13) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_PROT
	cause	
	DTI_CAUSE_NORMAL_CLOSE	
(14) DTI2_DISCONNECT_REQ		
	link_id	LINK_ID_PEER
	cause	
	DTI_CAUSE_NORMAL_CLOSE	
(15) PPP_TERMINATE_IND		
	ppp_cause	PPP_TERM_INSUF_RES

History:

18-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
03-Juli-2001	XOF	LCP TermAck corrected
15-Sept-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

4.6 Link Establishment Confirmation (PPP050-PPP059)

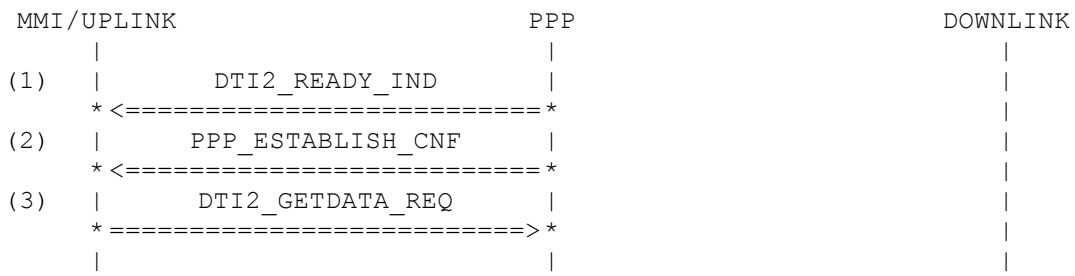
4.6.1 PPP050: Confiramtion of link establishment in client mode

Description:

PPP sends PPP_ESTABLISH_CNF after link establishment without Address-and-Control-Field-Compression and Protocol-Field-Compression in client mode.

Preamble:

PPP040A



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_READY_IND		
		link_id	LINK_ID_PROT
(2)	PPP_ESTABLISH_CNF		
		mru	PPP_MRU_DEFAULT
		ppp_hc	PPP_HC_OFF
		msid	MSID_DUMMY
		ip	IP_ADDR
		dns1	PPP_DNS1_DYNAMIC
		dns2	PPP_DNS2_DYNAMIC
(3)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.6.2 PPP051: Open Flow Control for Downlink

Description:

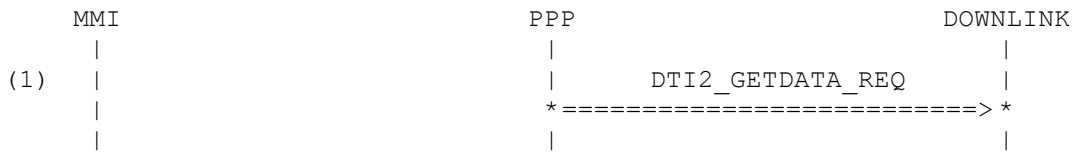
After receptin of an DTI2_GETDATA_REQ from UPLINK (PPP050) PPP must send an DTI2_GETDATA_REQ to the DOWNLINK entity.

Variants:

<A>....<B>

Preamble:

<A>PPP050
<B>PPP055



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

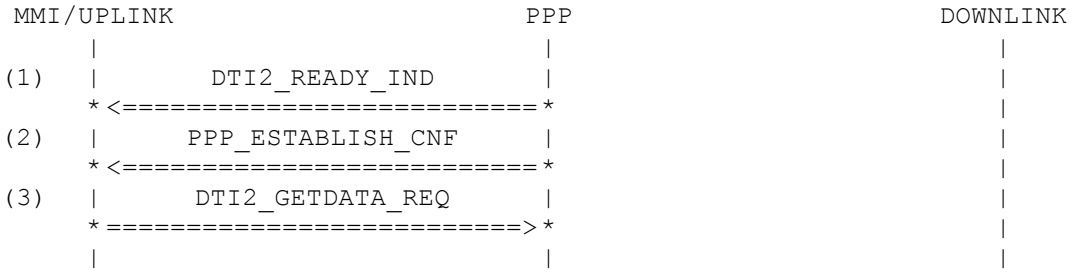
4.6.3 PPP052: Confiramtion of link establishment in client mode (ACFC, PFC)

Description:

PPP sends PPP_ESTABLISH_CNF after link establishment with Address-and-Control-Field-Compression and Protocol-Field-Compression in client mode.

Preamble:

PPP041



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND		
	link_id	LINK_ID_PROT
(2) PPP_ESTABLISH_CNF		
	mru	PPP_MRU_DEFAULT
	ppp_hc	PPP_HC_OFF
	msid	MSID_DUMMY
	ip	IP_ADDR
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

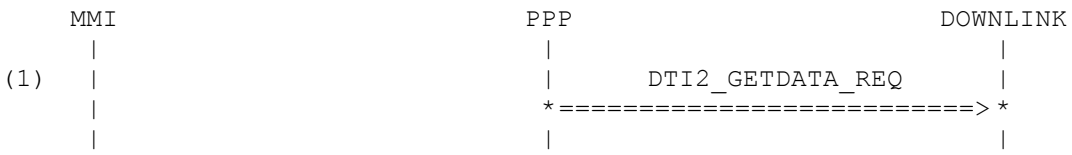
4.6.4 PPP053: Open Flow Control for Downlink (ACFC, PFC)

Description:

After receptin of an DTI2_GETDATA_REQ from UPLINK (PPP050) PPP must send an DTI2_GETDATA_REQ to the DOWNLINK entity.

Preamble:

PPP052



Parametrization:

	Primitive	Parameter	Value
(1)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

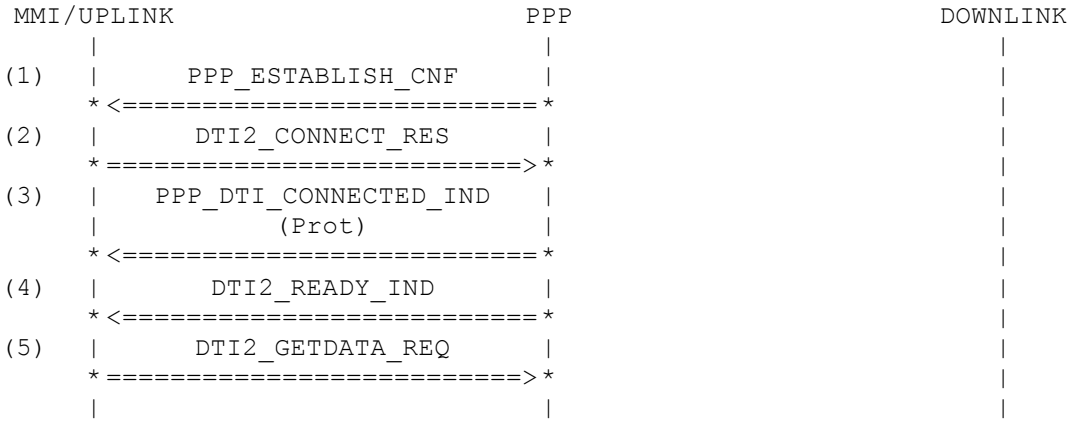
4.6.5 PPP055: Confirmation of link establishment in client mode and acknowledgement of prot dti connection

Description:

PPP sends PPP_ESTABLISH_CNF after link establishment without Address-and-Control-Field-Compression and Protocol-Field-Compression in client mode.

Preamble:

PPP040B



Parametrization:

	Primitive	Parameter	Value
(1)	PPP_ESTABLISH_CNF		
		mru	PPP_MRU_DEFAULT
		ppp_hc	PPP_HC_OFF
		msid	MSID_DUMMY
		ip	IP_ADDR
		dns1	PPP_DNS1_DYNAMIC
		dns2	PPP_DNS2_DYNAMIC
(2)	DTI2_CONNECT_RES		
		link_id	LINK_ID_PROT
		version	
		DTI_VERSION_10	

(3)	PPP_DTI_CONNECTED_IND		
		connected_direction	PPP_DTI_CONN_PROT
(4)	DTI2_READY_IND		
		link_id	LINK_ID_PROT
(5)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT

History:

26-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.7 Uplink Transfer (PPP060-PPP69)

4.7.1 PPP060: IP Packet in HDLC Frame

Description:

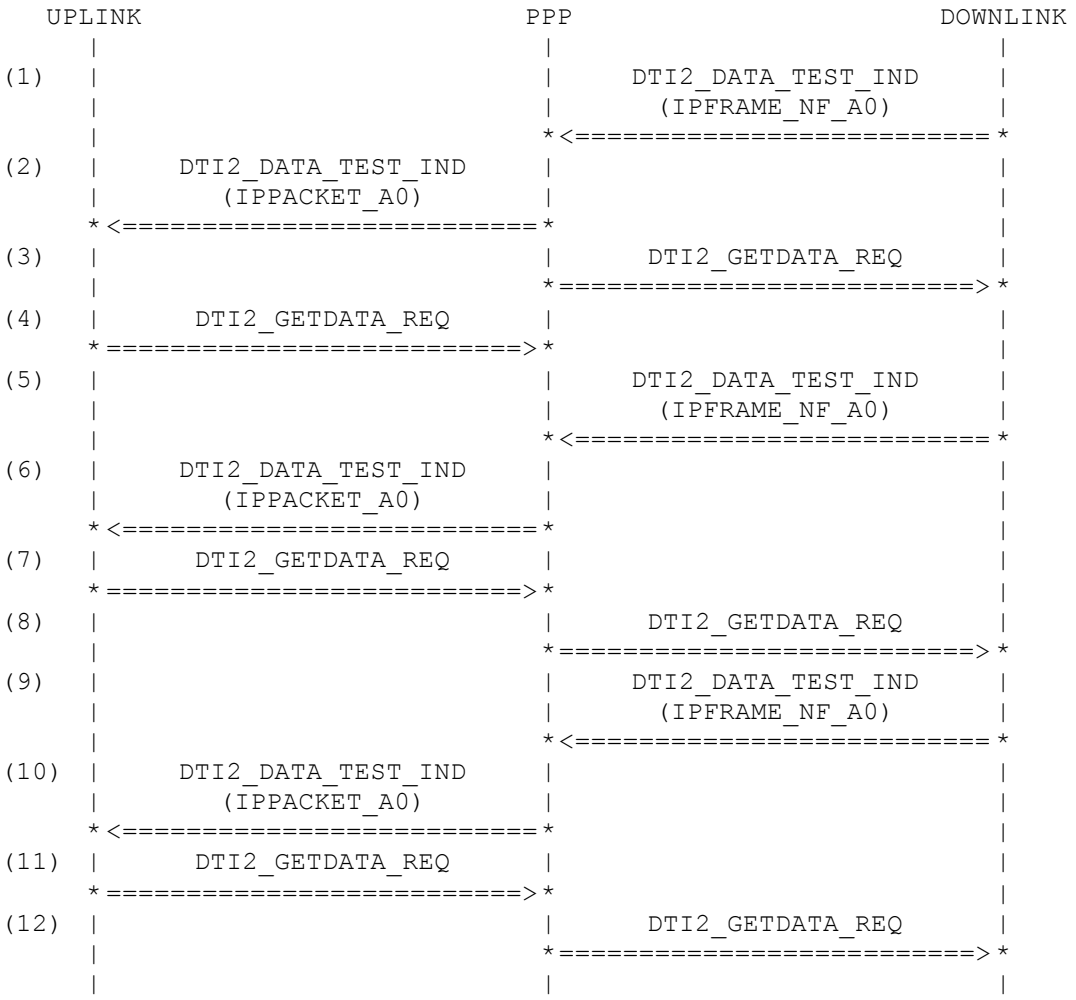
Convert an HDLC frame to an IP packet.

Variants:

<A>....<B>

Preamble:

<A>PPP051A
<B>PPP051B



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_NF_A0
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_NF_A0
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0

(7)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT
(8)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(9)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPFRAME_NF_A0
(10)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PROT
		parameters	DTI_PARAMETER_PROT
		sdu	IPPACKET_A0
(11)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT
(12)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow COntrol order

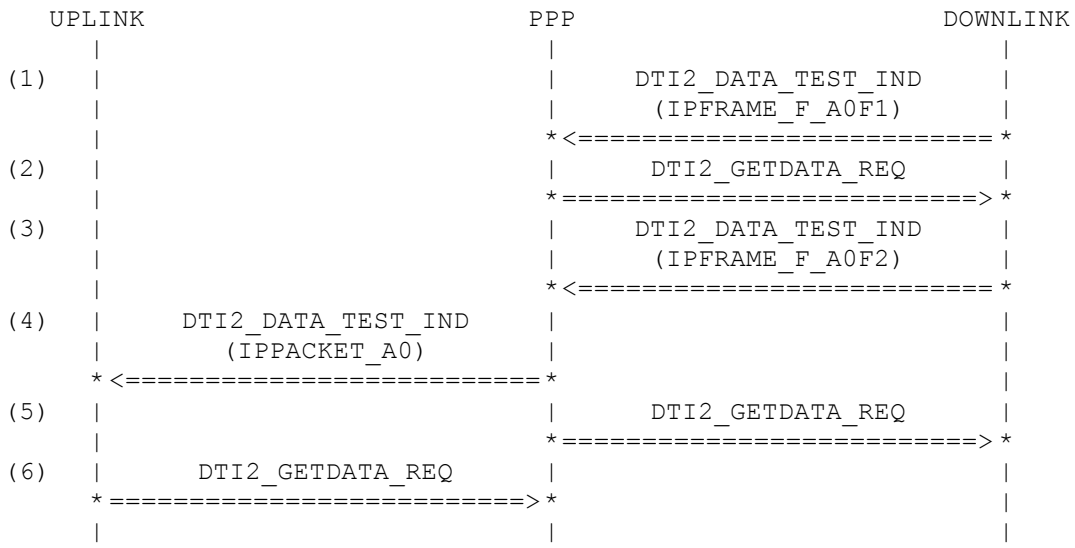
4.7.2 PPP061: IP Packet in two HDLC Frames

Description:

Convert two HDLC frames to one IP packet.

Preamble:

PPP051A



Parametrization:

Primitive	Parameter	Value

(1)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPFRAME_F_A0F1
(2)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(3)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPFRAME_F_A0F2
(4)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PROT
		parameters	DTI_PARAMETER_PROT
		sdu	IPPACKET_A0
(5)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(6)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

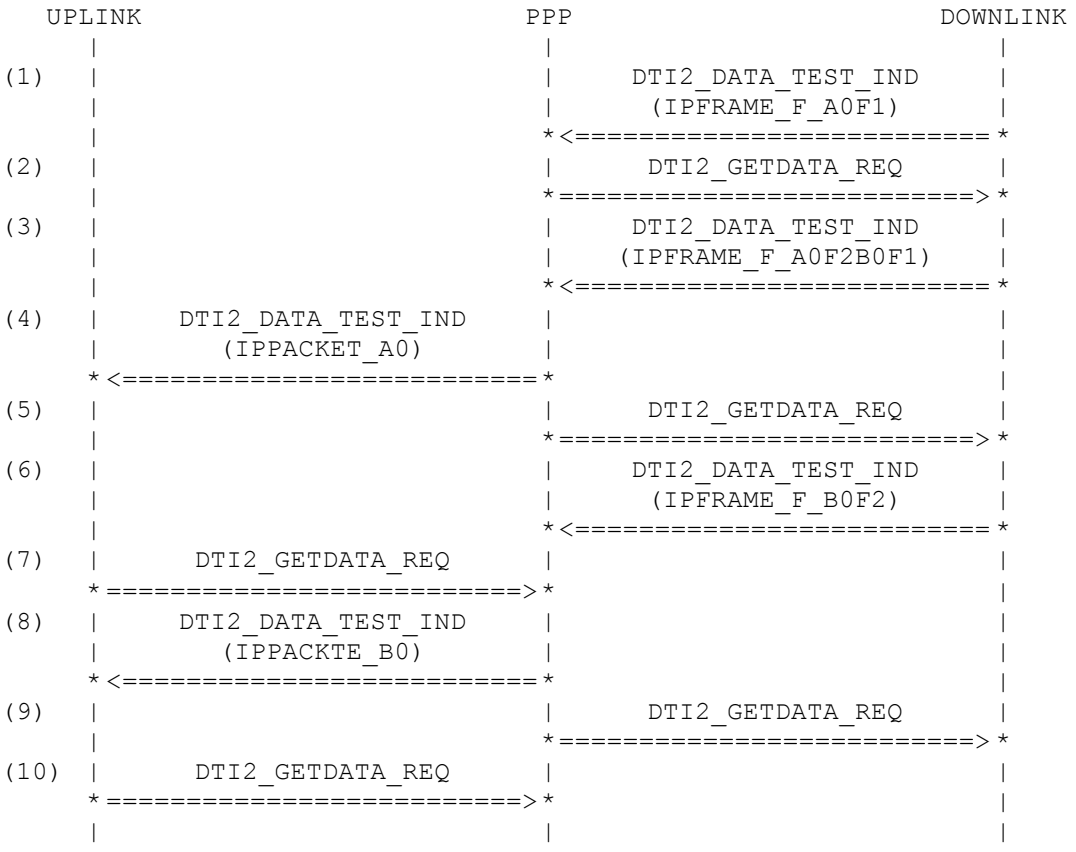
4.7.3 PPP062: Two Packets in three Frames

Description:

Convert three HDCL frames to two IP packets

Preamble:

PPP051A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_A0F1
(2) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_A0F2B0F1
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_B0F2
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT

	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_B0
(9) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(10) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.7.4 PPP063: IP Packet in HDLC Frame (ACFC, PFC)

Description:

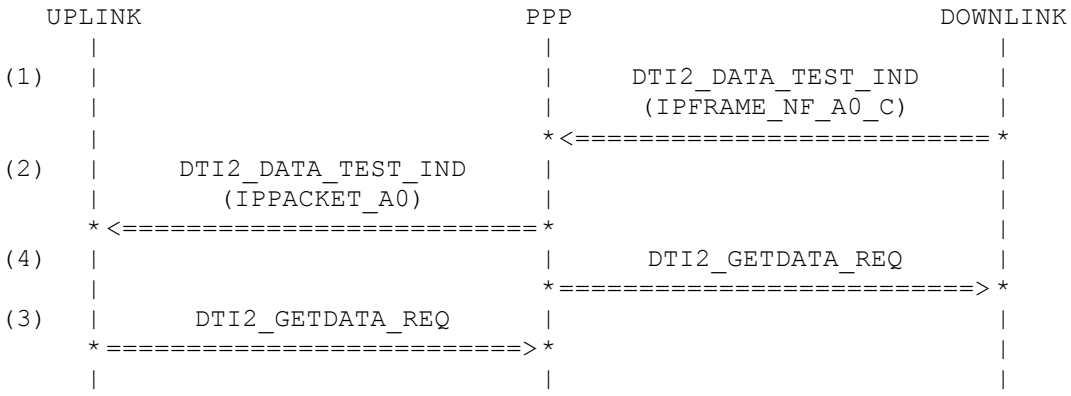
Convert HDLC frame with Address-and-Control-Field-Compression and Protocol-Field-Compression to an
IP packet

Variants:

<A>....<B>

Preamble:

<A>PPP043A
<B>PPP053



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_NF_A0_C
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

(4)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT

History:

17-Jun-2000	SG	Initial
17-Oct-2000	STW	add server mode preambles
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.7.5 PPP064: IP Packet in two HDLC Frames (ACFC, PFC)

Description:

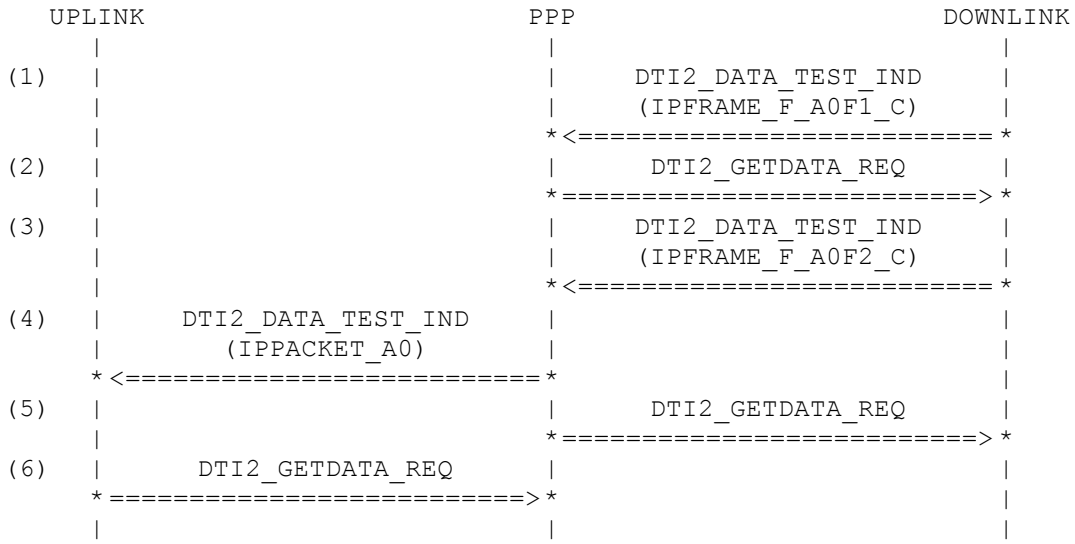
Convert two HDLC packts with Address-and-Control-Field-Compression and Protocol-Field-Compression to an IP packet.

Variants:

<A>....<D>

Preamble:

<A>PPP063A
<B>PPP063B
<C>PPP043A
<D>PPP053



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_A0F1_C
(2) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_A0F2_C

(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

17-Jun-2000	SG	Initial
17-Oct-2000	STW	add server mode preables
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.7.6 PPP065: Two Packets in three Frames (ACFC, PFC)

Description:

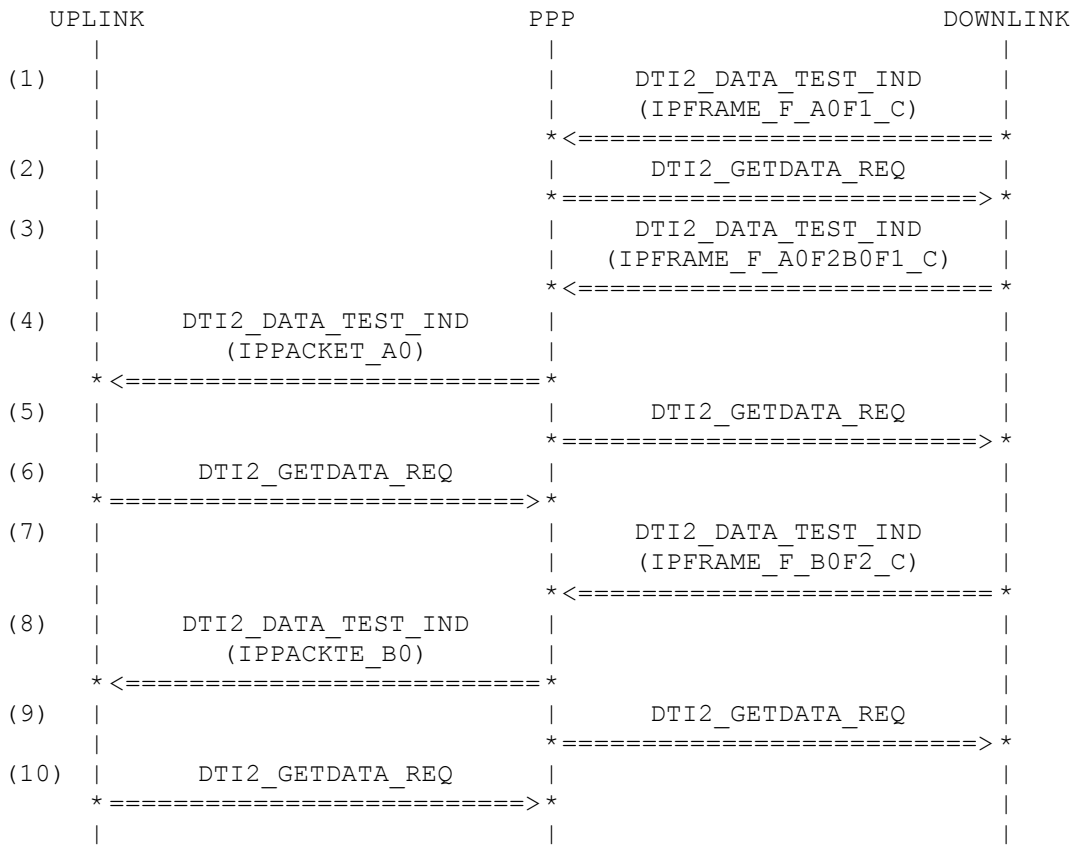
Convert three HDLC frames with Address-and-Control-Field-Compression and Protocol-Field-Compression to two IP packts.

Variants:

<A>....<F>

Preamble:

<A>PPP064A
<B>PPP064B
<C>PPP064C
<D>PPP064D
<E>PPP043A
<F>PPP053



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_A0F1_C
(2) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_A0F2B0F1_C
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(5) DTI2_GETDATA_REQ	link_id	LINK_ID_PEER
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(7) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_F_B0F2_C
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT

	parameters	DTI_PARAMETER_PROT
		sdu IPPACKET_B0
(9) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(10) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

17-Jun-2000	SG	Initial
17-Oct-2000	STW	add server mode preambles
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

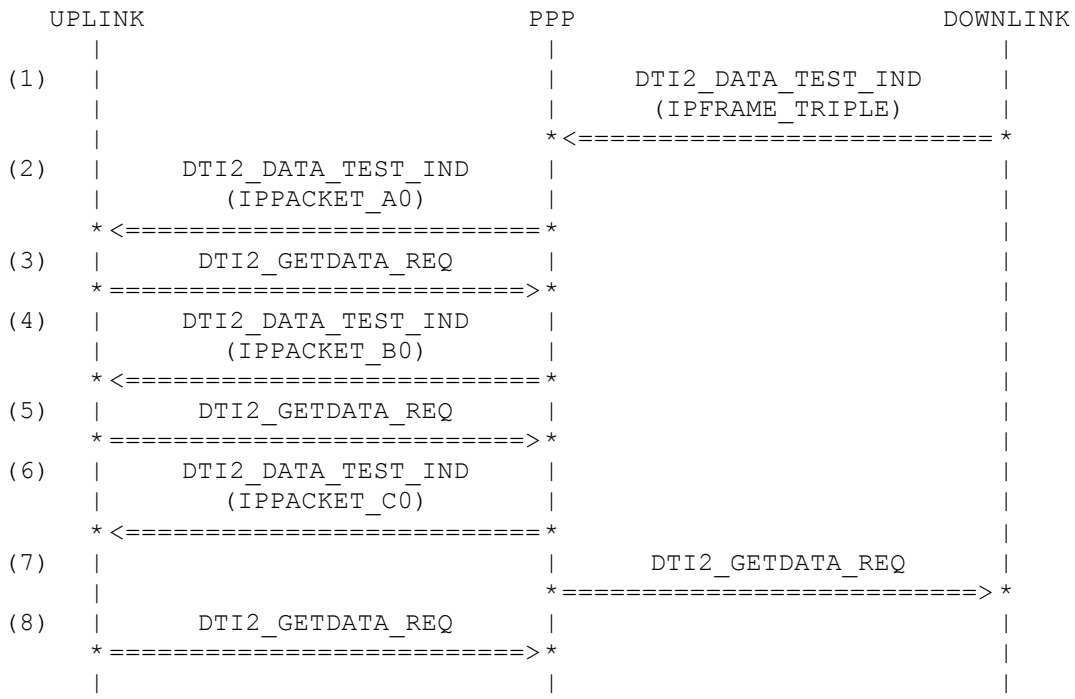
4.7.7 PPP066: Three Packets in one Frames

Description:

Convert one HDLC frame to three IP packts.

Preamble:

PPP051A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_TRIPLE
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT

	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_B0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_C0
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

20-Sep-2000	MT	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.7.8 PPP067: Three Packets in one Frame (ACFC,PFC)

Description:

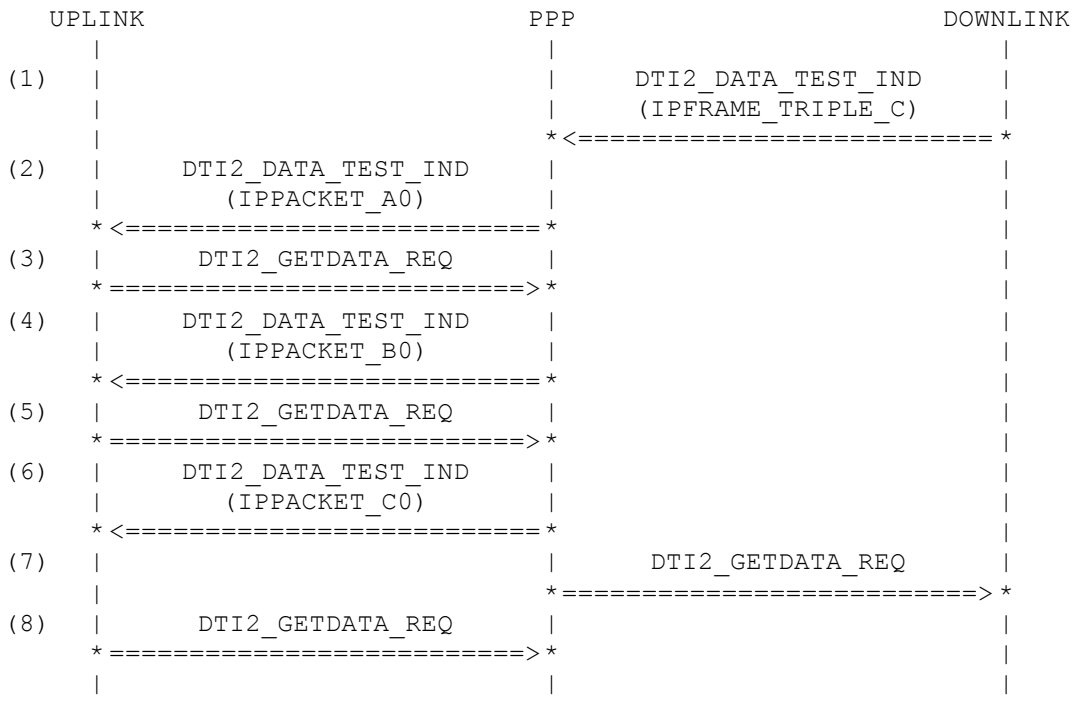
Convert one HDLC frame with Address-and-Control-Field-Compression and Protocol-Field-Compression to three IP packts.

Variants:

<A>....<H>

Preamble:

<A>PPP065A
<B>PPP065B
<C>PPP065C
<D>PPP065D
<E>PPP065E
<F>PPP065F
<G>PPP043A
<H>PPP053



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_TRIPLE_C
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_B0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_C0
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

(9) History:

20-Sep-2000	MT	Initial
17-Oct-2000	STW	add server mode preambles
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.7.9 PPP068: TCP Packets without Header Compression Negotiation (ACFC,PFC)

Description:

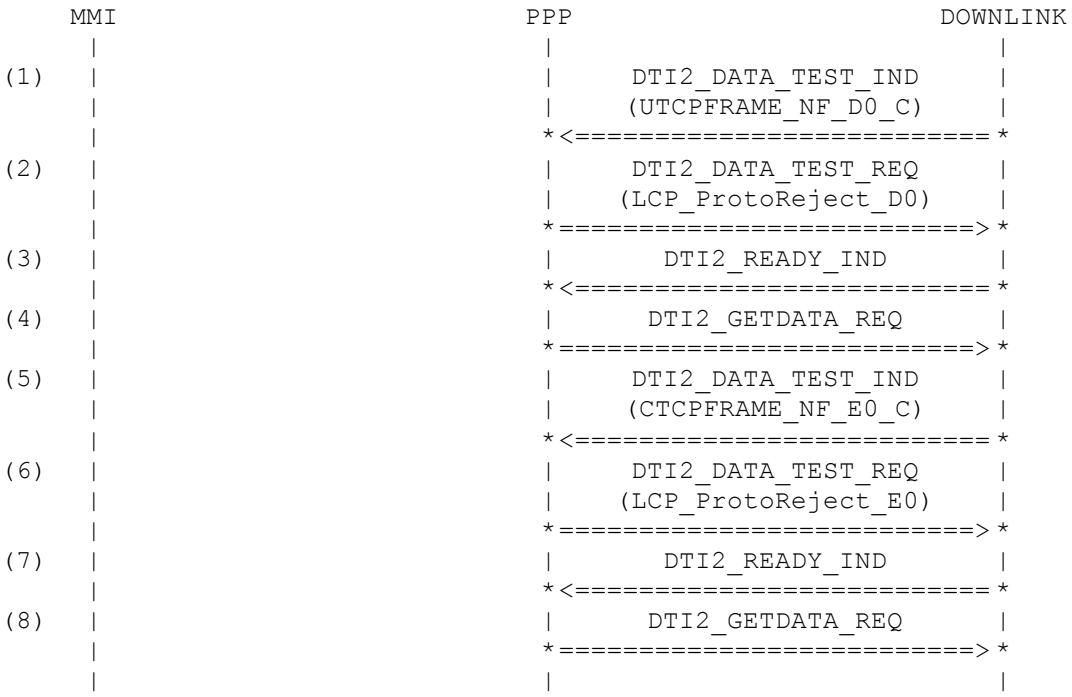
PPP peer sends compressed and uncompressed TCP packets, but Header Compression was not negotiated in IPCP. So these packets will be rejected by LCP-Protocol-Reject.

Variants:

<A>....<B>

Preamble:

<A>PPP067B
<B>PPP043E



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	UTCPFRAME_NF_D0_C
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ProtoReject_D0
<B>	sdu	LCP_ProtoReject_D0B
(3) DTI2_READY_IND		
	link_id	LINK_ID_PEER

(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	CTCPFRAME_NF_E0_C
(6) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ProtoReject_E0
<B>	sdu	LCP_ProtoReject_E0B
(7) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

19-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.7.10 PPP069: TCP Packets with Header Compression Negotiation (ACFC,PFC)

Description:

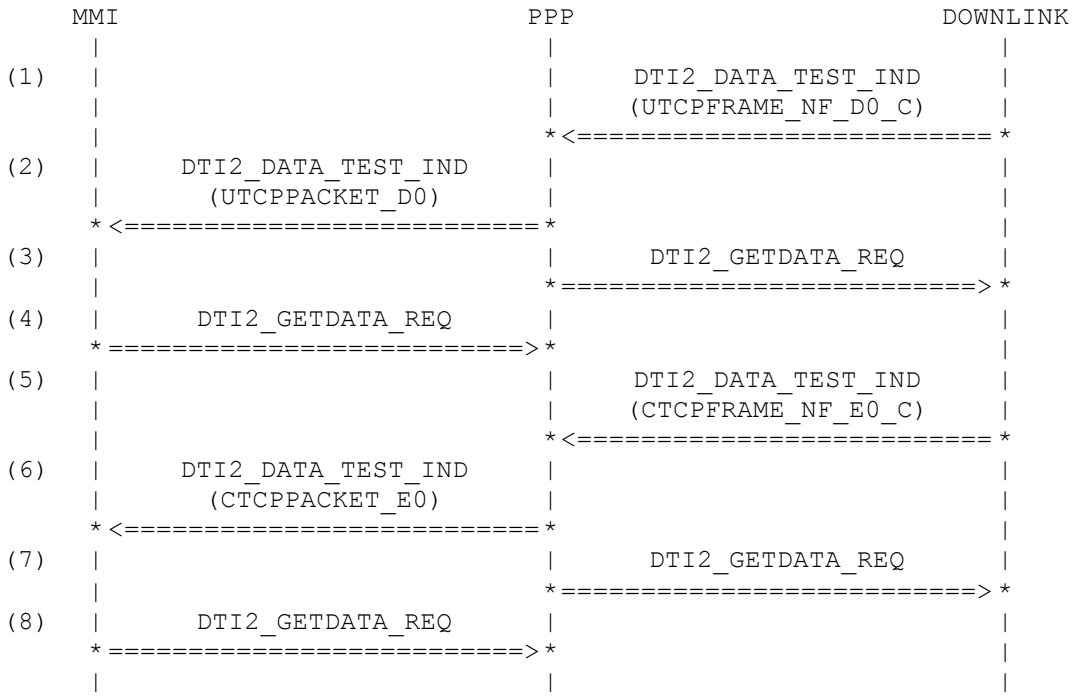
PPP peer sends compressed and uncompressed TCP packets. Header Compression was negotiated in IPCP so these packets will be pass to the upper layer.

Variants:

<A>….<B>

Preamble:

<A>PPP067A
<B>PPP043A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	UTCPFRAME_NF_D0_C
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT2
	sdu	UTCPPACKET_D0
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	CTCPFRAME_NF_E0_C
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT3
	sdu	CTCPPACKET_E0
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT

History:

20-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.8 Downlink Transfer (PPP070-PPP79)

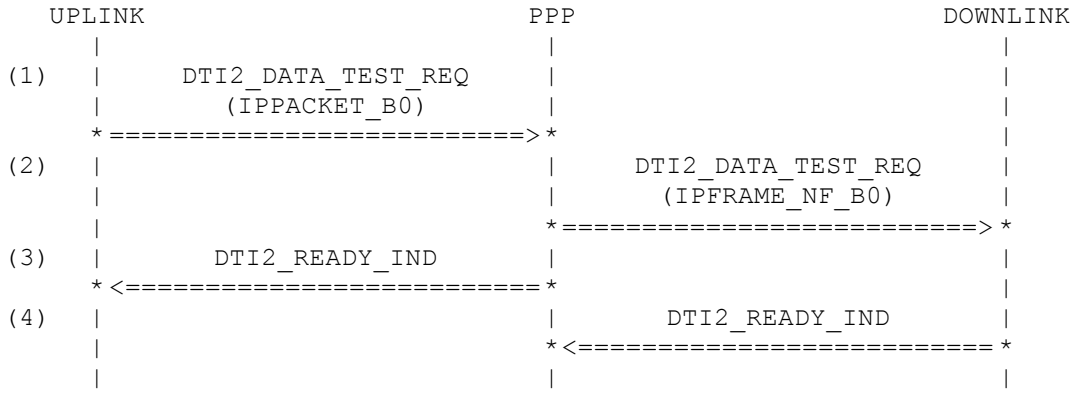
4.8.1 PPP070: Simple Downlink transfer

Description:

Convert an IP packt to a HDLC frame.

Preamble:

PPP051A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_B0
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_NF_B0
(3) DTI2_READY_IND		
	link_id	LINK_ID_PROT
(4) DTI2_READY_IND		
	link_id	LINK_ID_PEER

History:

17-Jun-2000	SG	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.8.2 PPP071: Simple Downlink transfer (ACFC, PFC)

Description:

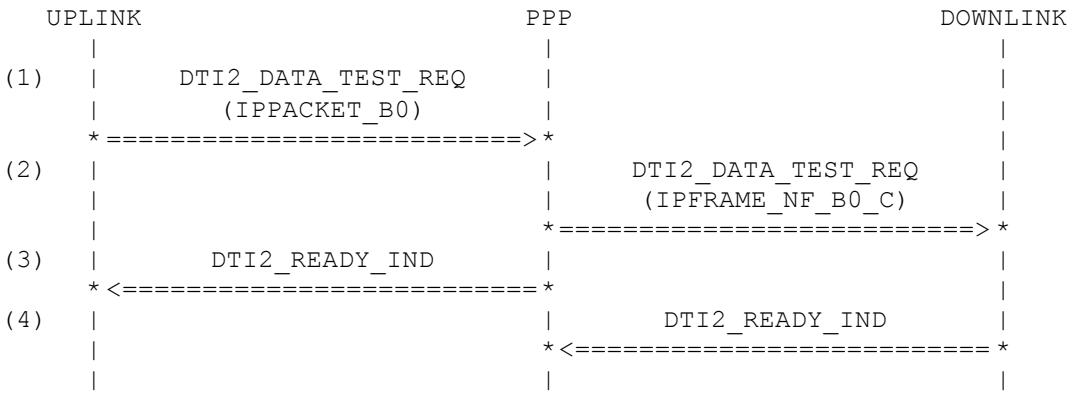
Convert an IP packet to a HDLC frame with Address-and-Control-Field-Compression and Protocol-Field-Compression

Variants:

<A>….<J>

Preamble:

<A>PPP067A
<B>PPP067B
<C>PPP067C
<D>PPP067D
<E>PPP067E
<F>PPP067F
<G>PPP067G
<H>PPP067H
<I>PPP043A
<J>PPP053



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_B0
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPFRAME_NF_B0_C
(3) DTI2_READY_IND		
	link_id	LINK_ID_PROT
(4) DTI2_READY_IND		
	link_id	LINK_ID_PEER

History:

17-Jun-2000	SG	Initial
17-Oct-2000	STW	add server mode preables
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.9 PPP Termination (PPP080-PPP083)

4.9.1 PPP080: Termination initiated by MMI, lower layer available

Description:

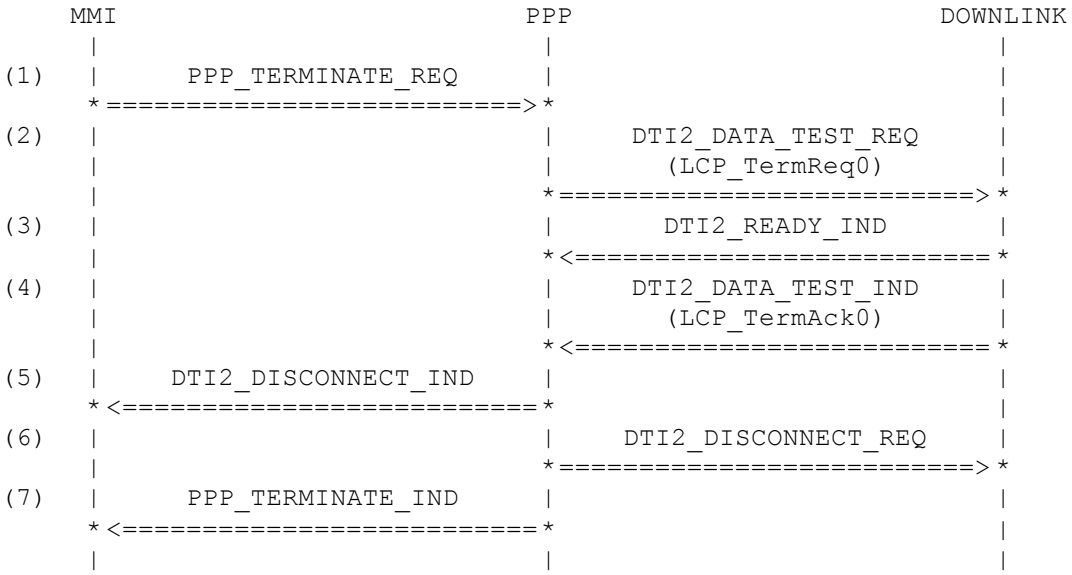
Successful termination of PPP while lower layer is available. PPP inform peer about termination.

Variants:

<A>....<C>

Preamble:

<A>PPP071A
<B>PPP071B
<C>PPP062



Parametrization:

Primitive	Parameter	Value
(1) PPP_TERMINATE_REQ		
	lower_layer	PPP_LOWER_LAYER_UP
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermReq0
(3) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER

		parameters	DTI_PARAMETER_PEER
		sdu	LCP_TermAck0
(5)	DTI2_DISCONNECT_IND		
		link_id	LINK_ID_PROT
		cause	
		DTI_CAUSE_NORMAL_CLOSE	
(6)	DTI2_DISCONNECT_REQ		
		link_id	LINK_ID_PEER
		cause	
		DTI_CAUSE_NORMAL_CLOSE	
(7)	PPP_TERMINATE_IND		
		ppp_cause	PPP_TERM_OK_MMI

History:

17-Jun-2000	SG	Initial
18-Oct-2000	STW	add server mode preambles
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

4.9.2 PPP081: Termination initiated by MMI, lower layer not available

Description:

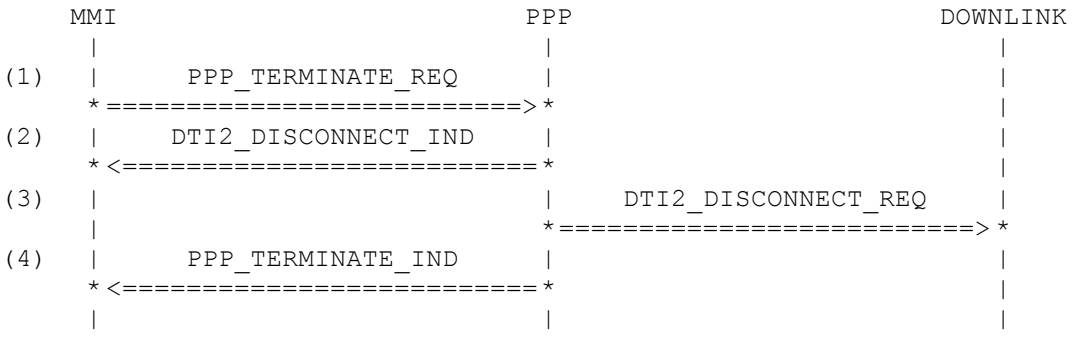
Successful termination of PPP while lower layer is not available.

Variants:

<A>....<C>

Preamble:

<A>PPP071A
<B>PPP071B
<C>PPP062



Parametrization:

Primitive	Parameter	Value
(1) PPP_TERMINATE_REQ		
	lower_layer	PPP_LOWER_LAYER_DOWN
(2) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_PROT
	cause	
	DTI_CAUSE_NORMAL_CLOSE	

(3)	DTI2_DISCONNECT_REQ		
		link_id	LINK_ID_PEER
		cause	
		DTI_CAUSE_NORMAL_CLOSE	
(4)	PPP_TERMINATE_IND		
		ppp_cause	PPP_TERM_OK_MMI

History:

17-Jun-2000	SG	Initial
18-Oct-2000	STW	add server mode preambles
15-Sep-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

4.9.3 PPP082: Termination initiated by PPP Peer

Description:

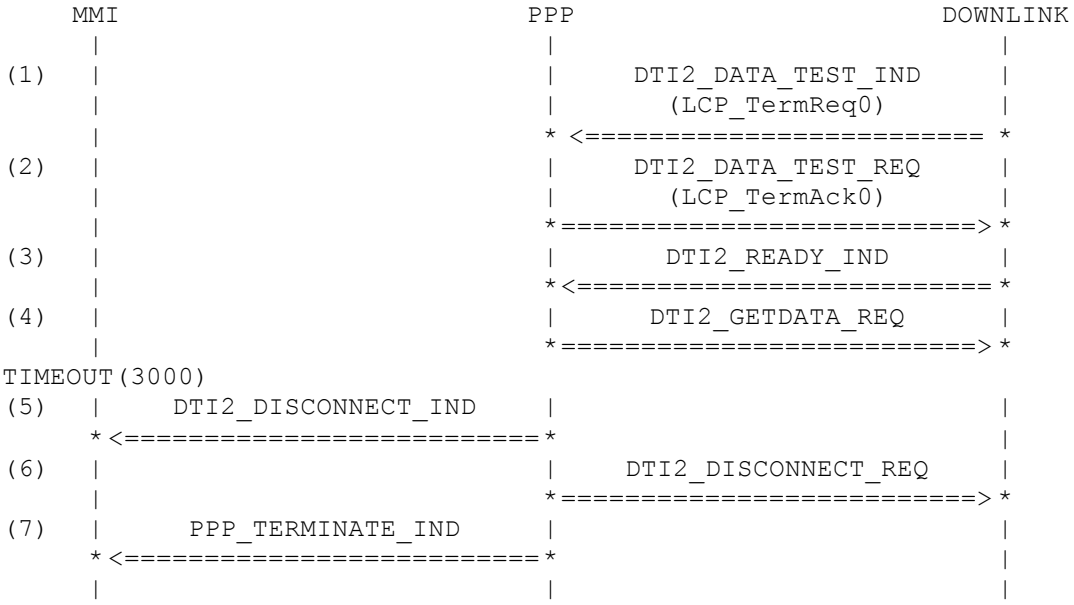
PPP peer desire to terminate the PPP link, PPP informs MMI that the PPP link is terminated.

Variants:

<A>....<C>

Preamble:

<A>PPP071A
<B>PPP071B
<C>PPP062



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermReq0
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER

	parameters	DTI_PARAMETER_PEER
	sdu	LCP_TermAck0
(3) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(5) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_PROT
	cause	DTI_CAUSE_NORMAL_CLOSE
(6) DTI2_DISCONNECT_REQ		
	link_id	LINK_ID_PEER
	cause	DTI_CAUSE_NORMAL_CLOSE
(7) PPP_TERMINATE_IND		
	ppp_cause	PPP_TERM_OK_PEER

History:

17-Jun-2000	SG	Initial
18-Oct-2000	STW	add server mode preambles
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

4.10 Uplink Transfer with invalid packets (PPP090-PPP099)

4.10.1 PPP090: one frame with invalid control field received

Description:

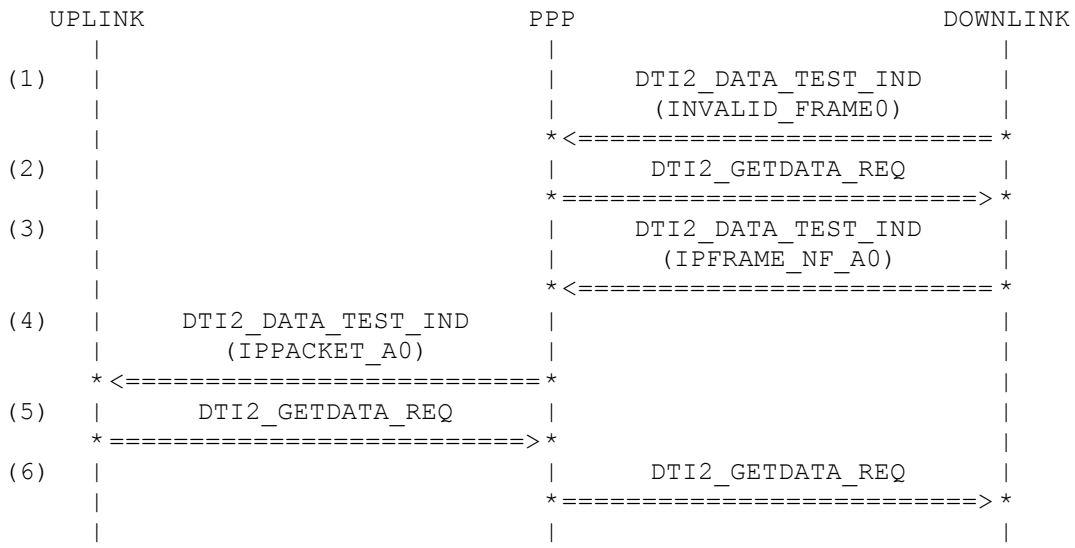
PPP discards invalid packet and sends “get data request” immediately.

Variants:

<A>….<D>

Preamble:

<A>PPP071A
<B>PPP071B
<C>PPP043A
<D>PPP060A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	INVALID_FRAME0
(2) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPFRAME_NF_A0_C
<B>	sdu	IPFRAME_NF_A0
<C>	sdu	IPFRAME_NF_A0_C
<D>	sdu	IPFRAME_NF_A0
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

19-Sep-2000	MT	Initial
18-Oct-2000	STW	add server mode primitives
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality

4.10.2 PPP091: one frame with invalid protocol field received

Description:

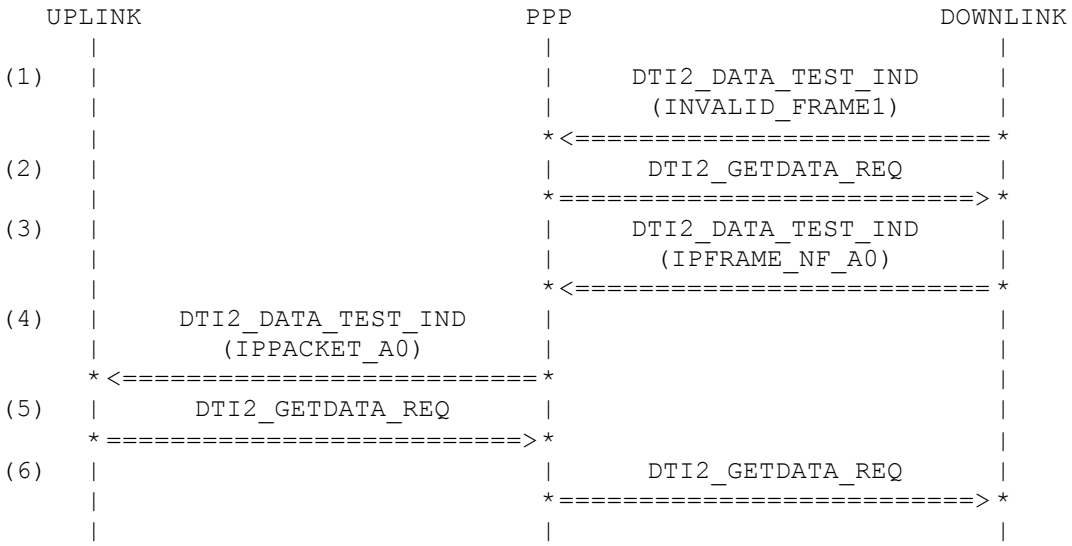
PPP discards invalid packet and sends “get data request” immediately

Variants:

<A>….<D>

Preamble:

<A>PPP090A
<B>PPP090B
<C>PPP043A
<D>PPP060A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	INVALID_FRAME1
(2) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPFRAME_NF_A0_C
<B>	sdu	IPFRAME_NF_A0
<C>	sdu	IPFRAME_NF_A0_C
<D>	sdu	IPFRAME_NF_A0
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

19-Sep-2000	MT	Initial
18-Oct-2000	STW	add server mode primitives
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality

4.10.3 PPP092: one frame with invalid address field received

Description:

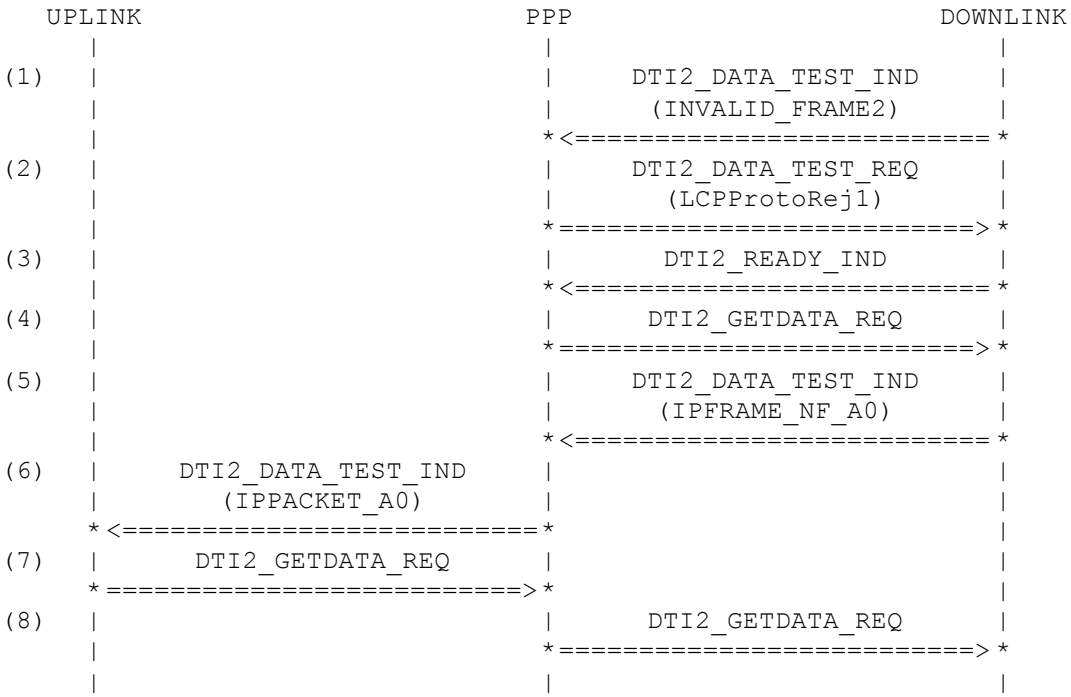
PPP sends LCP protocol reject

Variants:

<A>....<D>

Preamble:

<A>PPP091A
<B>PPP091B
<C>PPP043A
<D>PPP060A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	INVALID_FRAME2
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	LCP_ProtoReject88
<B>	sdu	LCPProtoRej1

<C>	sdu	LCP_ProtoReject88
<D>	sdu	LCPProtoRej1
(3) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(4) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(5) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPFRAME_NF_A0_C
<B>	sdu	IPFRAME_NF_A0
<C>	sdu	IPFRAME_NF_A0_C
<D>	sdu	IPFRAME_NF_A0
(6) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(8) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

20-Sep-2000	MT	Initial
18-Oct-2000	STW	add server mode primitves
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality

4.10.4 PPP093: one frame with invalid FSC received

Description:

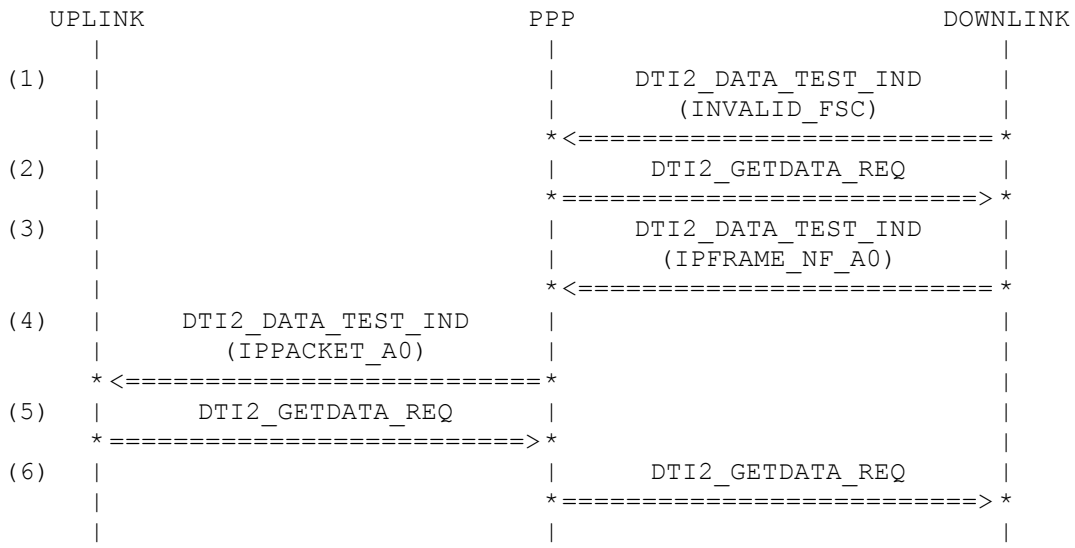
PPP silently discards invalid packet and sends “get data request” immediately

Variants:

<A>….<D>

Preamble:

<A>PPP092A
<B>PPP092B
<C>PPP043A
<D>PPP060A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	INVALID_FSC
(2) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(3) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
<A>	sdu	IPFRAME_NF_A0_C
<B>	sdu	IPFRAME_NF_A0
<C>	sdu	IPFRAME_NF_A0_C
<D>	sdu	IPFRAME_NF_A0
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PROT
	sdu	IPPACKET_A0
(5) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

20-Sep-2000	MT	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.11 Modification of PPP link (PPP100-PPP109)

4.11.1 PPP100: turn header compression off

Description:

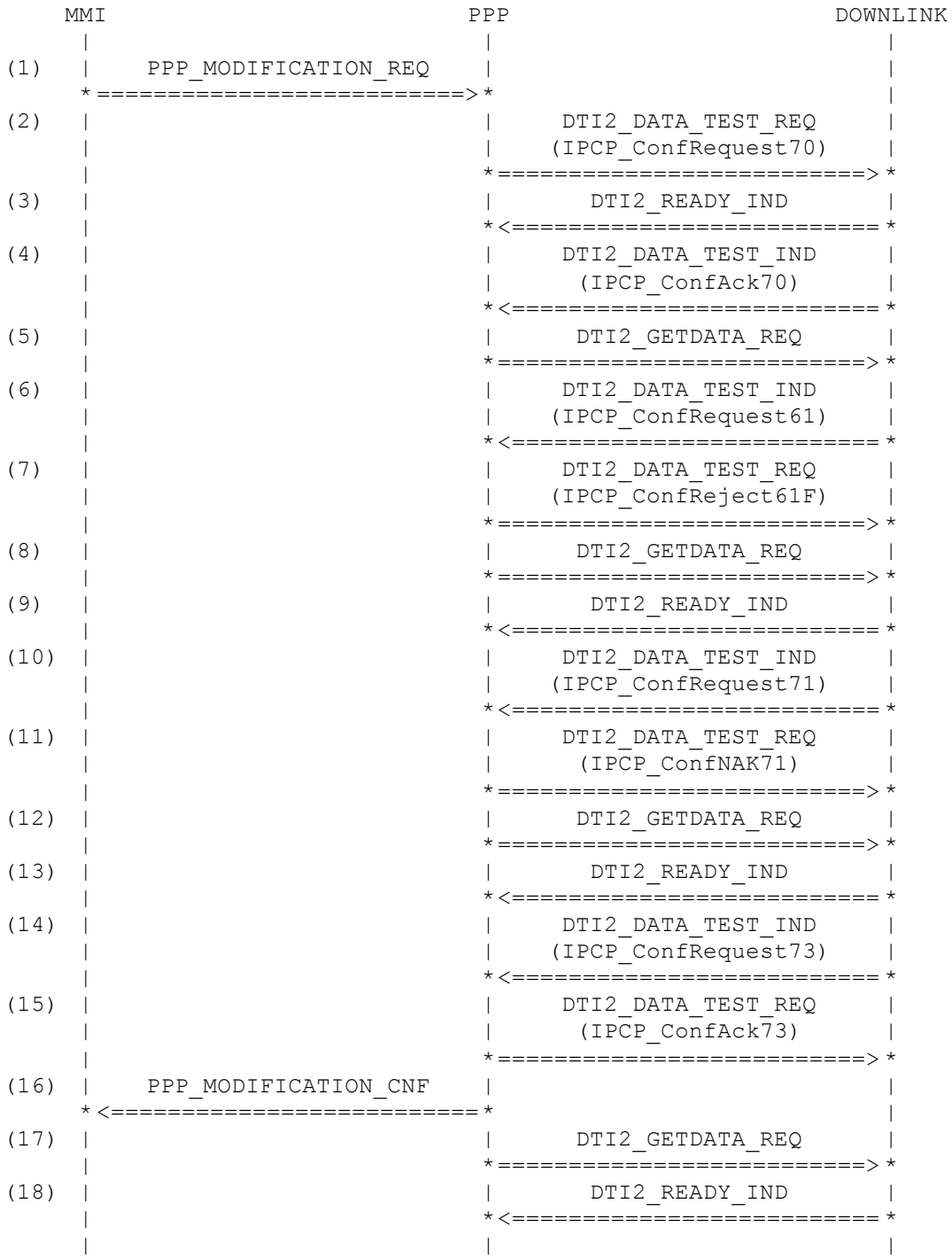
PPP initiates a new IPCP negotiation which turns off header compression.

Variants:

<A>….<B>

Preamble:

<A>PPP093A
<B>PPP043A



Parametrization:

Primitive	Parameter	Value
(1) PPP_MODIFICATION_REQ		
	ppp_hc	PPP_HC_OFF
	msid	MSID_DUMMY
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfRequest88

(3) DTI2_READY_IND

link_id	LINK_ID_PEER

(4) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfAck88

(5) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(6) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfRequest61

(7) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfReject61F

(8) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(9) DTI2_READY_IND

link_id	LINK_ID_PEER

(10) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfRequest71

(11) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfNAK71

(12) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(13) DTI2_READY_IND

link_id	LINK_ID_PEER

(14) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfRequest73

(15) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfAck73

(16) PPP_MODIFICATION_CNF

ppp_hc	PPP_HC_OFF
msid	MSID_DUMMY

(17) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(18) DTI2_READY_IND

link_id	LINK_ID_PEER

History:

18-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

4.12 Transparent packet transfer (PPP110-PPP119)

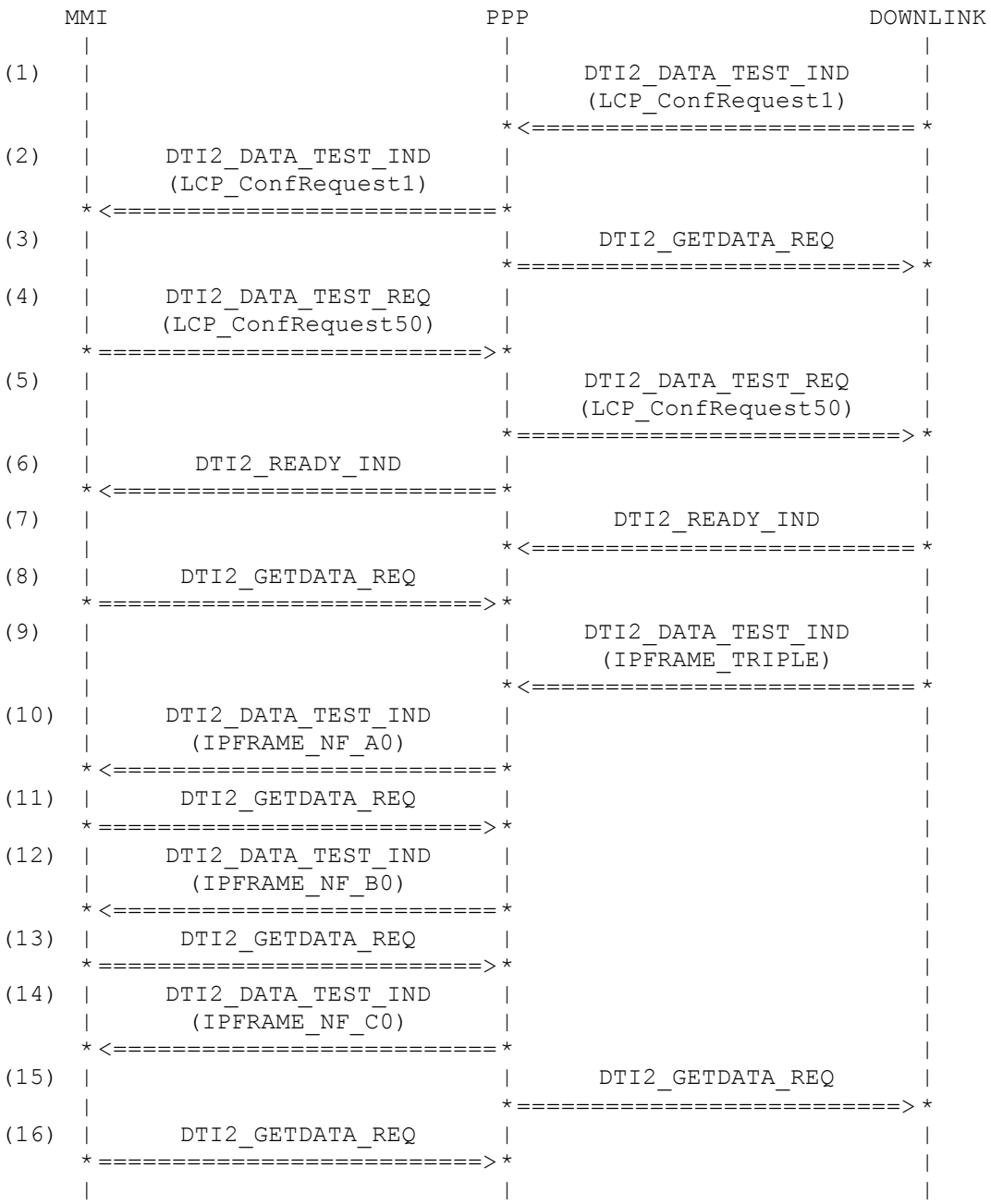
4.12.1 PPP110: Transparent packet transfer uplink and downlink

Description:

PPP packet transfer uplink and downlink.

Preamble:

PPP014



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest1
(2) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PROT
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest1
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

(4) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PROT
parameters	DTI_PARAMETER_PEER
sdu	LCP_ConfRequest50

(5) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	LCP_ConfRequest50

(6) DTI2_READY_IND

link_id	LINK_ID_PROT

(7) DTI2_READY_IND

link_id	LINK_ID_PEER

(8) DTI2_GETDATA_REQ

link_id	LINK_ID_PROT

(9) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPFRAME_TRIPLE

(10) DTI2_DATA_TEST_IND

link_id	LINK_ID_PROT
parameters	DTI_PARAMETER_PEER
sdu	IPFRAME_NF_A0

(11) DTI2_GETDATA_REQ

link_id	LINK_ID_PROT

(12) DTI2_DATA_TEST_IND

link_id	LINK_ID_PROT
parameters	DTI_PARAMETER_PEER
sdu	IPFRAME_NF_B0

(13) DTI2_GETDATA_REQ

link_id	LINK_ID_PROT

(14) DTI2_DATA_TEST_IND

link_id	LINK_ID_PROT
parameters	DTI_PARAMETER_PEER
sdu	IPFRAME_NF_C0

(15) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(16) DTI2_GETDATA_REQ

link_id	LINK_ID_PROT

History:

18-Oct-2000	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order

4.13 Link termination in transparent mode

4.13.1 PPP120: Termination of transparent link

Description:

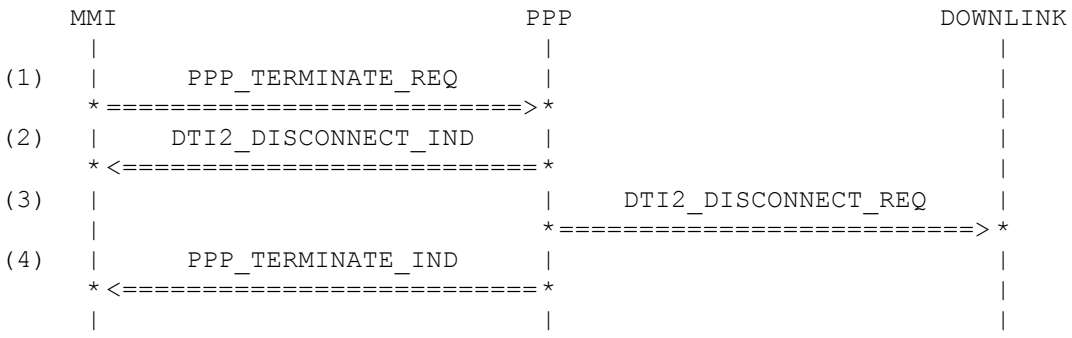
Termination of PPP link in transparent mode.

Variants:

<A>....<B>

Preamble:

<A>PPP014
<B>PPP110



Parametrization:

Primitive	Parameter	Value
(1) PPP_TERMINATE_REQ		
	lower_layer	PPP_LOWER_LAYER_DOWN
(2) DTI2_DISCONNECT_IND		
	link_id	LINK_ID_PROT
	cause	DTI_CAUSE_NORMAL_CLOSE
(3) DTI2_DISCONNECT_REQ		
	link_id	LINK_ID_PEER
	cause	DTI_CAUSE_NORMAL_CLOSE
(4) PPP_TERMINATE_IND		
	ppp_cause	PPP_TERM_OK_MMI

History:

23-Oct-2000	STW	Initial
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

4.14 Link reestablishment

4.14.1 PPP130: Reestablishment in client mode

Description:

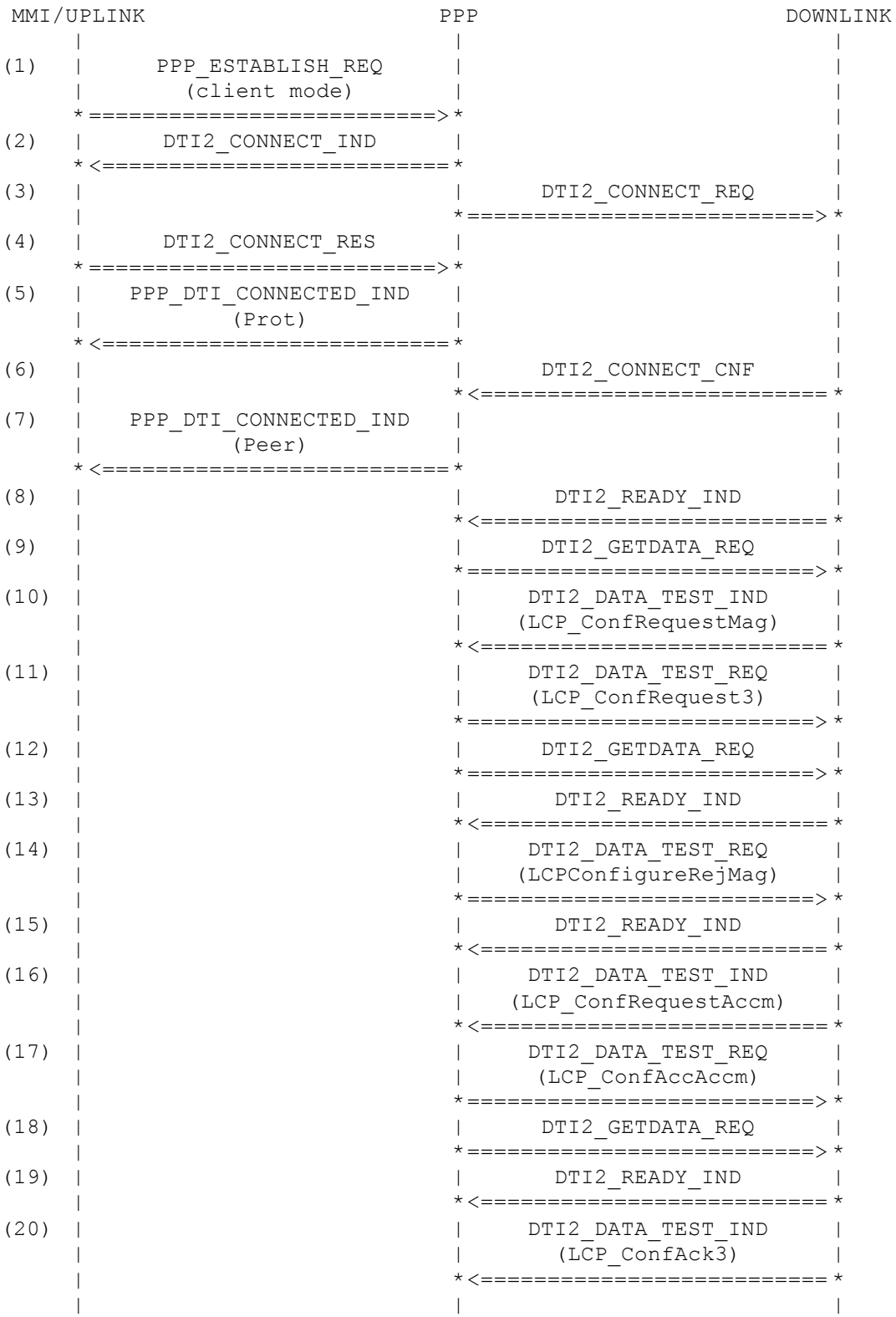
Shows Reestablishment in client mode with PAP authentification, ACFC (Address-and-Control-Field-Compression) am PFC (Protocol-Field-Compression). Magic number is rejected by PPP client.

Variants:

<A>....<C>

Preamble:

<A>PPP082A
<B>PPP082B
<C>PPP082C



Parametrization:

Primitive	Parameter	Value
(1) PPP_ESTABLISH_REQ		
	mode	PPP_CLIENT
	mru	PPP_MRU_DEFAULT
	ap	PPP_AP_PAP

	login	AUTH_STRUCT0
	accm	PPP_ACCM_DEFAULT
	rt	PPP_RT_DEFAULT
	mc	PPP_MC_DEFAULT
	mt	PPP_MT_DEFAULT
	mf	PPP_MF_DEFAULT
	ppp_hc	PPP_HC_OFF
	ip	PPP_IP_DYNAMIC
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
	peer_channel	PEER_CHANNEL0
	protocol_channel	PROTOCOL_CHANNEL0
	peer_direction DTI_CHANNEL_TO_LOWER_LAYER	
	prot_direction DTI_CHANNEL_TO_HIGHER_LAYER	
	peer_link_id	LINK_ID_PEER
	prot_link_id	LINK_ID_PROT
(2) DTI2_CONNECT_IND		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(3) DTI2_CONNECT_REQ		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(4) DTI2_CONNECT_RES		
	link_id	LINK_ID_PROT
	version	DTI_VERSION_10
(5) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PROT
(6) DTI2_CONNECT_CNF		
	link_id	LINK_ID_PEER
	version	DTI_VERSION_10
(7) PPP_DTI_CONNECTED_IND		
	connected_direction	PPP_DTI_CONN_PEER
(8) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(9) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(10) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequestMag
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ConfRequest3
(12) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(13) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(14) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER

		parameters	DTI_PARAMETER_PEER
		sdu	LCPConfigureRejMag
(15)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(16)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfRequestAccm
(17)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAccAccm
(18)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(19)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(20)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	LCP_ConfAck3

History:

19-Jan-2001	STW	Initial
22-Jun-2001	XOF	Channel ID added
12-Jul-2001	STW	Change primitve order
17-Sept-2001	XOF	Change to DTILIB

4.15 IP address negotiation (special cases)

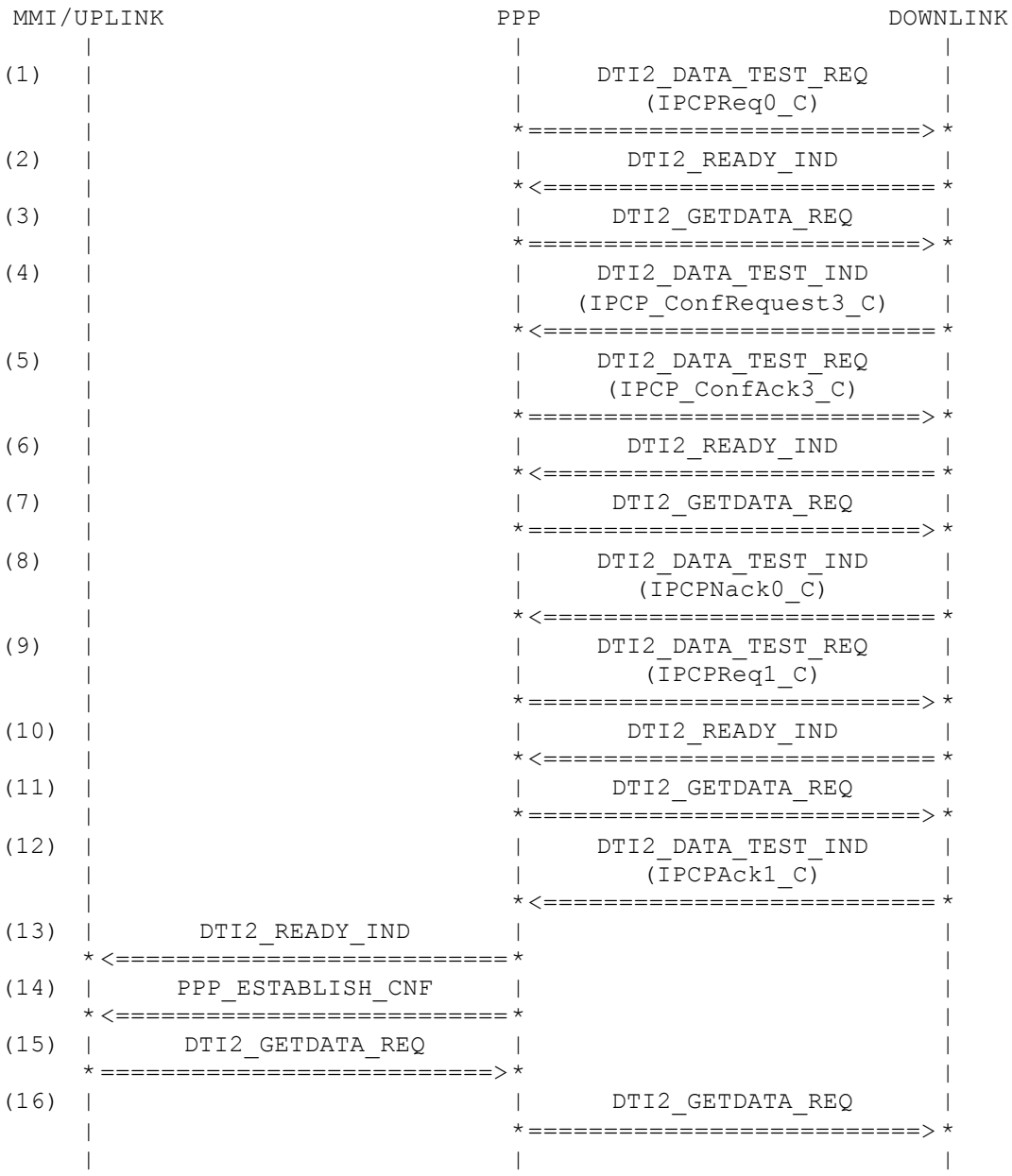
4.15.1 PPP140: Server IP address in client mode allowed

Description:

If the server sends its IP address in client mode PPP acknowledged it.

Preamble:

PPP031



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq0_C
(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfRequest3_C

(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfAck3_C
(6) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPNack0_C
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq1_C
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck1_C
(13) DTI2_READY_IND		
	link_id	LINK_ID_PROT
(14) PPP_ESTABLISH_CNF		
	mru	PPP_MRU_DEFAULT
	ppp_hc	PPP_HC_OFF
	msid	MSID_DUMMY
	ip	IP_ADDR
	dns1	PPP_DNS1_DYNAMIC
	dns2	PPP_DNS2_DYNAMIC
(15) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PROT
(16) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

History:

19-Jan-2001	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

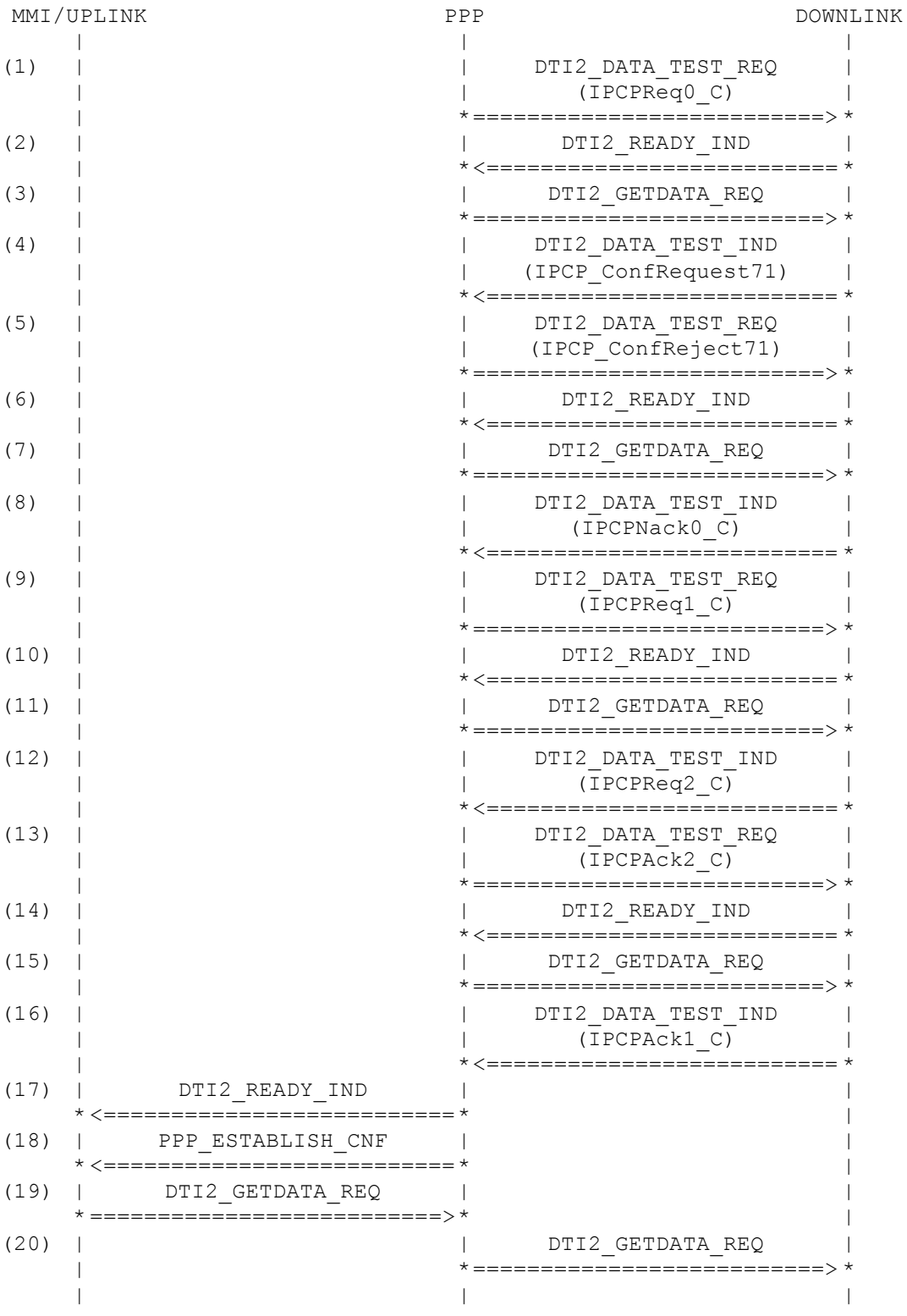
4.15.2 PPP141: Rejection of dynamic server IP address in client mode

Description:

If the server sends a dynamic server IP address in client mode PPP must reject it.

Preamble:

PPP031



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq0_C

(2) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfRequest71
(5) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfReject71
(6) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(7) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(8) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPNack0_C
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq1_C
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPReq2_C
(13) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck2_C
(14) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(15) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(16) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCPAck1_C
(17) DTI2_READY_IND		
	link_id	LINK_ID_PROT
(18) PPP_ESTABLISH_CNF		
	mru	PPP_MRU_DEFAULT
	ppp_hc	PPP_HC_OFF
	msid	MSID_DUMMY
	ip	IP_ADDR

		dns1	PPP_DNS1_DYNAMIC
		dns2	PPP_DNS2_DYNAMIC
(19)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT
(20)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER

History:

19-Jan-2001	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sept-2001	XOF	Build DTILIB functionality

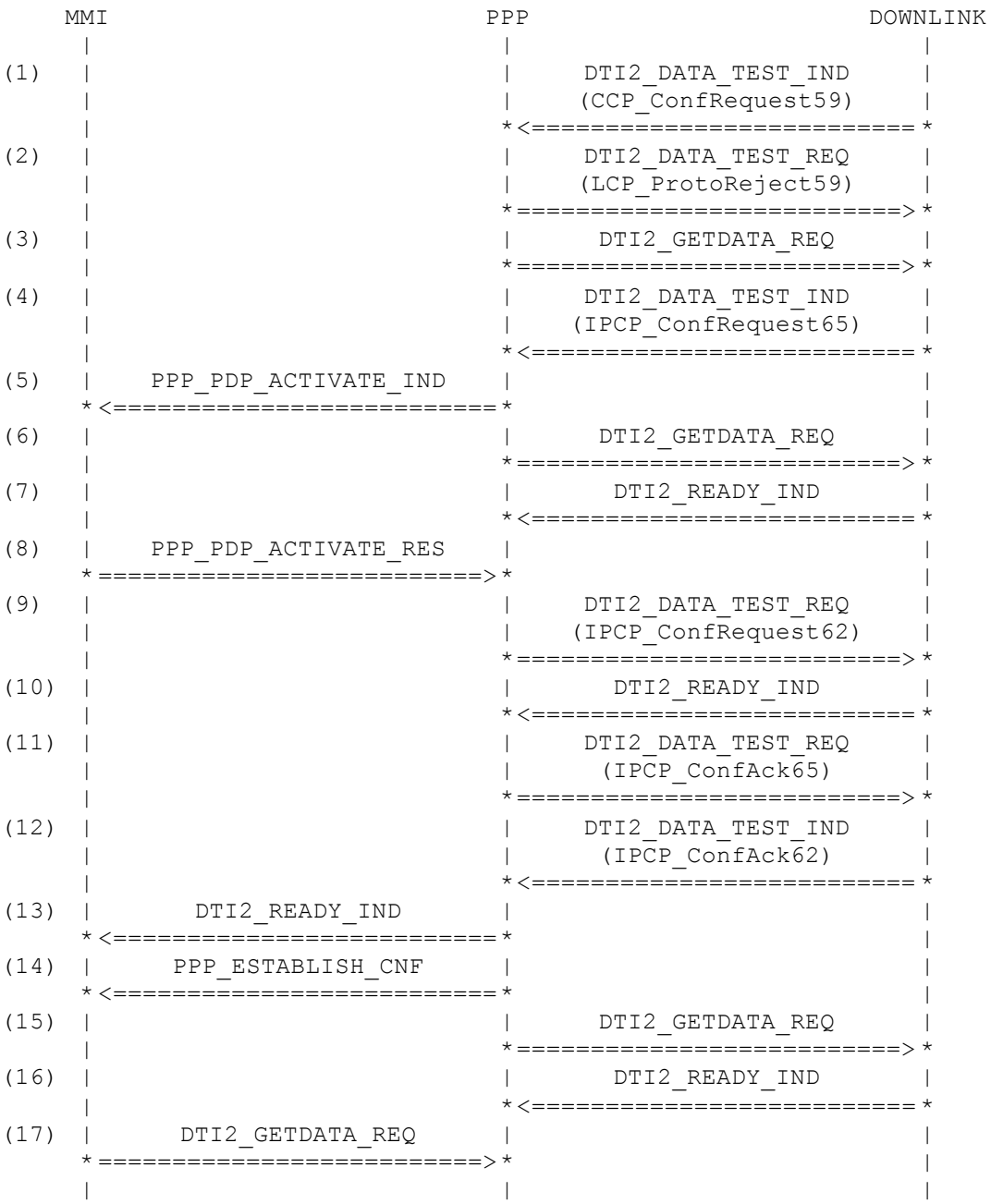
4.15.3 PPP142: Allow always static DNS addresses in server mode

Description:

IPCP configuration in server mode. If the Network does not respond with Protocol Configuration Options (PCO) PPP ackowledges static DNS addresses. PPP receives a CCP (Compression Control Protocol) configure request and sends a LCP protocol reject. The IPCP negotiation procedure continues.

Preamble:

PPP033A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	CCP_ConfRequest59
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ProtoReject59
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER

(4) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfRequest65

(5) PPP_PDP_ACTIVATE_IND

ppp_hc	PPP_HC_VJ
msid	MSID_EXPECTED
sdu	
PCO_LIST_IND_CHAP_STATIC	

(6) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(7) DTI2_READY_IND

link_id	LINK_ID_PEER

(8) PPP_PDP_ACTIVATE_RES

ppp_hc	PPP_HC_VJ
msid	MSID_EXPECTED
ip	PPP_IP_141_64_21_2
sdu	PCO_LIST_RES_EMPTY

(9) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfRequest62

(10) DTI2_READY_IND

link_id	LINK_ID_PEER

(11) DTI2_DATA_TEST_REQ

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfAck65

(12) DTI2_DATA_TEST_IND

link_id	LINK_ID_PEER
parameters	DTI_PARAMETER_PEER
sdu	IPCP_ConfAck62

(13) DTI2_READY_IND

link_id	LINK_ID_PROT

(14) PPP_ESTABLISH_CNF

mru	PPP_MRU_DEFAULT
ppp_hc	PPP_HC_VJ
msid	MSID_EXPECTED
ip	PPP_IP_141_64_21_2
dns1	PPP_PDNS_141_64_24_129
dns2	PPP_SDNS_141_64_254_131

(15) DTI2_GETDATA_REQ

link_id	LINK_ID_PEER

(16) DTI2_READY_IND

link_id	LINK_ID_PEER

(17) DTI2_GETDATA_REQ

link_id	LINK_ID_PROT

History:

19-Jan-2001	STW	Initial
22-Juni-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality
25-Oct-2002	STW	Cause concept

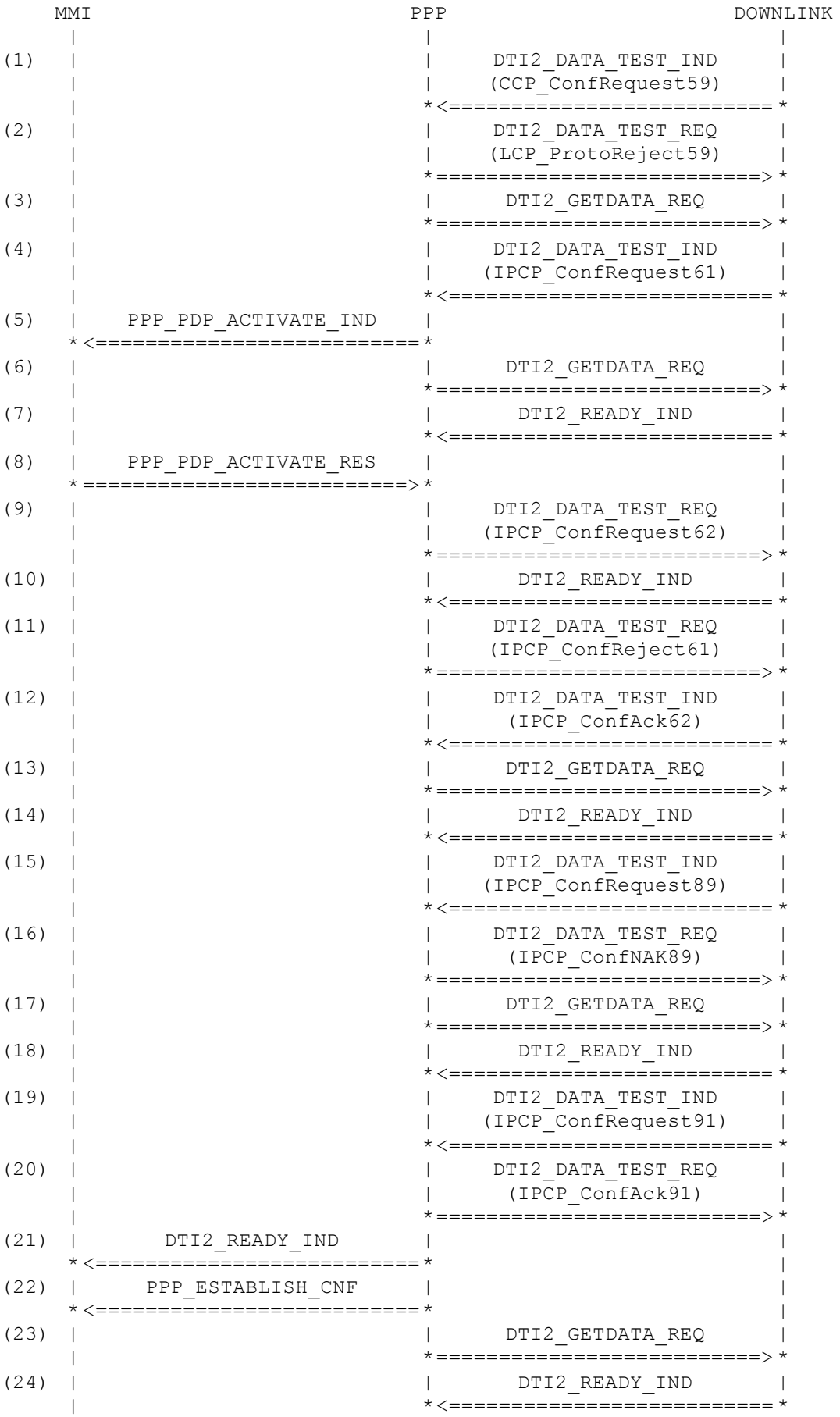
4.15.4 PPP143: Reject dynamic DNS addresses in server mode if network does not answer

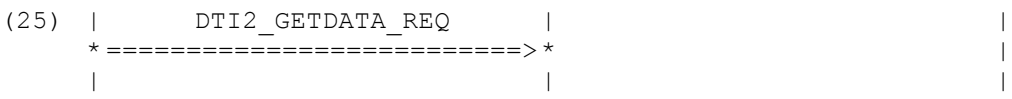
Description:

IPCP configuration in server mode. If the Network does not respond with Protocol Configuration Options (PCO) PPP rejects dynamic DNS addresses. PPP receives a CCP (Compression Control Protocol) configure request and sends a LCP protocol reject. The IPCP negotiation procedure continues.

Preamble:

PPP033A


MMI
PPP
DOWNLINK
(1)
DTI2_DATA_TEST_IND
(CCP_ConfRequest59)
(2)
DTI2_DATA_TEST_REQ
(LCP_ProtoReject59)
(3)
DTI2_GETDATA_REQ
(4)
DTI2_DATA_TEST_IND
(IPCP_ConfRequest61)
(5)
PPP_PDP_ACTIVATE_IND
(6)
DTI2_GETDATA_REQ
(7)
DTI2_READY_IND
(8)
PPP_PDP_ACTIVATE_RES
(9)
DTI2_DATA_TEST_REQ
(IPCP_ConfRequest62)
(10)
DTI2_READY_IND
(11)
DTI2_DATA_TEST_REQ
(IPCP_ConfReject61)
(12)
DTI2_DATA_TEST_IND
(IPCP_ConfAck62)
(13)
DTI2_GETDATA_REQ
(14)
DTI2_READY_IND
(15)
DTI2_DATA_TEST_IND
(IPCP_ConfRequest89)
(16)
DTI2_DATA_TEST_REQ
(IPCP_ConfNAK89)
(17)
DTI2_GETDATA_REQ
(18)
DTI2_READY_IND
(19)
DTI2_DATA_TEST_IND
(IPCP_ConfRequest91)
(20)
DTI2_DATA_TEST_REQ
(IPCP_ConfAck91)
(21)
DTI2_READY_IND
(22)
PPP_ESTABLISH_CNF
(23)
DTI2_GETDATA_REQ
(24)
DTI2_READY_IND



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	CCP_ConfRequest59
(2) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	LCP_ProtoReject59
(3) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(4) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfRequest61
(5) PPP_PDP_ACTIVATE_IND		
	ppp_hc	PPP_HC_VJ
	msid	MSID_EXPECTED
	sdu	PCO_LIST_IND_CHAP
(6) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(7) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(8) PPP_PDP_ACTIVATE_RES		
	ppp_hc	PPP_HC_VJ
	msid	MSID_EXPECTED
	ip	PPP_IP_141_64_21_2
	sdu	PCO_LIST_RES_EMPTY
(9) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfRequest62
(10) DTI2_READY_IND		
	link_id	LINK_ID_PEER
(11) DTI2_DATA_TEST_REQ		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfReject61DNS
(12) DTI2_DATA_TEST_IND		
	link_id	LINK_ID_PEER
	parameters	DTI_PARAMETER_PEER
	sdu	IPCP_ConfAck62
(13) DTI2_GETDATA_REQ		
	link_id	LINK_ID_PEER
(14) DTI2_READY_IND		
	link_id	LINK_ID_PEER

(15)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPCP_ConfRequest89
(16)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPCP_ConfNAK89
(17)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(18)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(19)	DTI2_DATA_TEST_IND		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPCP_ConfRequest91
(20)	DTI2_DATA_TEST_REQ		
		link_id	LINK_ID_PEER
		parameters	DTI_PARAMETER_PEER
		sdu	IPCP_ConfAck91
(21)	DTI2_READY_IND		
		link_id	LINK_ID_PROT
(22)	PPP_ESTABLISH_CNF		
		mru	PPP_MRU_DEFAULT
		ppp_hc	PPP_HC_VJ
		msid	MSID_EXPECTED
		ip	PPP_IP_141_64_21_2
		dns1	PPP_PDNS_DYNAMIC
		dns2	PPP_SDNS_DYNAMIC
(23)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PEER
(24)	DTI2_READY_IND		
		link_id	LINK_ID_PEER
(25)	DTI2_GETDATA_REQ		
		link_id	LINK_ID_PROT

History:

22-Jan-2001	STW	Initial
22-Jun-2001	XOF	Channel ID added
15-Sep-2001	XOF	Build DTILIB functionality
28-Jan-2002	STW	Change Flow Control order
25-Oct-2002	STW	Cause concept

Appendices

A. Acronyms

DS-WCDMA	Direct Sequence/Spread Wideband Code Division Multiple Access

B. Glossary

International Mobile Telecommunication 2000 (IMT-2000/ITU-2000)	Formerly referred to as FPLMTS (Future Public Land-Mobile Telephone System), this is the ITU's specification/family of standards for 3G. This initiative provides a global infrastructure through both satellite and terrestrial systems, for fixed and mobile phone users. The family of standards is a framework comprising a mix/blend of systems providing global roaming. <URL: http://www.imt-2000.org/>