Frame Relay IP Header Compression Implementation Agreement

FRF.20

Frame Relay Forum Technical Committee
June 2001

Note: The user's attention is called to the possibility that implementation of the Frame Relay implementation agreement contained herein may require the use of inventions covered by patent rights held by third parties. By publication of this Frame Relay implementation agreement, the Frame Relay Forum makes no representation that the implementation of the specification will not infringe on any third party rights. The Frame Relay Forum takes no position with respect to any claim that has been or may be asserted by any third party, the validity of any patent rights related to any such claims, or the extent to which a license to use any such rights may not be available.

Editor:

George Wilkie
Cisco Systems
96 Commercial Street
Edinburgh, EH6 6LX, United Kingdom
Phone: +44 131 561 3607
E-Mail: gwilkie@cisco.com

For more information contact:

The Frame Relay Forum
Suite 307
39355 California Street
Fremont, CA 94538 USA

Phone: +1 (510) 608-5920
FAX: +1 (510) 608-5917
E-Mail: frf@frforum.com
WWW: http://www.frforum.com

Table of Contents

List of Tables

List of Figures

Revision History

Version	Change	Date
FRF.iphc	Document approved as FRF.20.	June 2001

This Page Left Blank Intentionally

1 INTRODUCTION

1.1 Purpose

This document is a Frame Relay IP header compression implementation agreement.

The agreements herein were reached in the Frame Relay Forum, and are based on the relevant Frame Relay standards referenced in Section 1.4. They address the optional parts of these standards, and document agreements reached among vendors/suppliers of Frame Relay network products and services regarding the options to be implemented.

This document may be submitted to different bodies involved in ratification of implementation agreements and conformance testing to facilitate multi–vendor interoperability.

1.2 Definitions

Must, **Shall**, or **Mandatory** — the item is an absolute requirement of the implementation agreement.

Should — the item is highly desirable.

May or **Optional** — the item is not compulsory and may be followed or ignored according to the needs of the implementer.

1.3 Acronym List

CID	Context IDentifier
DCE	Data Circuit-terminating Equipment
DLCI	Data Link Connection Identifier
DTE	Data Terminal Equipment
FRIHCP	Frame Relay IP Header Compression control Protocol
FRPP	Frame Relay Privacy Protocol
IP	Internet Protocol
IPCP	IP Control Protocol
IPv4	Version 4 of the Internet Protocol
IPv6	Version 6 of the Internet Protocol
IPv6CP	IPv6 Control Protocol
IWF	Inter Working Function
LCP	Link Control Protocol
NCP	Network Control Protocol
NLPID	Network Layer Protocol IDentifier
PDU	Protocol Data Unit
PPP	Point to Point Protocol
PVC	Permanent Virtual Circuit
RTP	Real Time Protocol
SVC	Switched Virtual Circuit
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VC	Virtual Circuit

1.4 Relevant Standards

The following is a list of standards on which this implementation agreement is based:

[1] FRF.1.2, D. Sinicrope (ed.), User-to-Network Implementation Agreement (UNI), Frame Relay Forum, January 19, 2000.

[2] FRF.3.2, A. Malis (ed.), Multiprotocol Encapsulation Implementation Agreement, Frame Relay Forum, April 2000.

[3] FRF.4.1, D. Sinicrope (ed.), SVC User-to-Network Interface (UNI) Implementation Agreement, Frame Relay Forum, January 21, 2000.

[4] FRF.5, D. O'Leary (ed.), Frame Relay/ATM PVC Network Interworking Implementation Agreement, Frame Relay Forum, December 20, 1994.

[5] FRF.8.1, D. O'Leary (ed.), Frame Relay/ATM PVC Service Interworking Implementation Agreement, Frame Relay Forum, February 2000.

[6] FRF.9, D. Cantwell (ed.), Data Compression Over Frame Relay Implementation Agreement, Frame Relay Forum, January 22, 1996.

[7] FRF.11.1, T. Hatala *et al* (ed.), Voice over Frame Relay Implementation Agreement, Frame Relay Forum, December 1998.

[8] FRF.12, A. Malis (ed.), Frame Relay Fragmentation Implementation Agreement, Frame Relay Forum, December 1997.

[9] FRF.14, K. Rehbehn (ed.), Physical Layer Interface Implementation Agreement, Frame Relay Forum, December 1998.

[10] FRF.17, D. Sinicrope (ed.), Frame Relay Privacy Implementation Agreement, Frame Relay Forum, January 21, 2000.

[11] ITU-T Recommendation Q.922, ISDN Data Link Layer Specification for Frame Mode Bearer Services, ITU, Geneva, 1992.

[12] ITU-T Recommendation Q.921, ISDN User-Network Interface-Data Link Layer Specification, ITU, Geneva, 1997.

[13] ITU-T Recommendation Q.933, ISDN DSS 1 - Signaling Specifications for Frame Mode Switched and Permanent Virtual Connection Control and Status Monitoring, ITU, Geneva, 1995.

[14] ITU-T Recommendation Q.931, ISDN User-Network Interface Layer 3 Specification for Basic Call Control, ITU, Geneva, 1998.

[15] RFC 1144, V. Jacobson, Compressing TCP/IP Headers for Low-Speed Serial Links, IETF, February 1990.

[16] RFC 1332, G. McGregor, The PPP Internet Protocol Control Protocol (IPCP), IETF, May 1992.

[17] RFC 1661/STD 51, W. Simpson (ed.), PPP Link Control Protocol, IETF, July 1994.

[18] RFC 2472, D. Haskin & E. Allen, IP Version 6 over PPP, IETF, December 1998.

[19] RFC 2507, M. Degermark *et al*, IP Header Compression, IETF, February 1999.

[20] RFC 2508, S. Casner & V. Jacobson, Compressing IP/UDP/RTP Headers for Low-Speed Serial Links, IETF, February 1999.

[21] RFC 2509, M. Engan *et al*, IP Header Compression over PPP, IETF, February 1999.

2 IMPLEMENTATION AGREEMENTS

The remainder of this document contains the agreements reached in the Frame Relay Forum.

2.1 Overview

This document describes how compressed IP datagrams are encapsulated within a Frame Relay frame, and how IP header compression algorithms and their parameters are negotiated over Frame Relay VCs. The document applies to both IPv4 and IPv6.

Frame Relay IP Header Compression is based on the following Frame Relay Forum implementation agreements:

- FRF.9, Data Compression
- FRF.17, Frame Relay Privacy.

A Frame Relay IP Header Compression control Protocol (FRIHCP) is defined which negotiates the use of IP header compression in each direction of the VC. FRIHCP is based on the Link Control Protocol (RFC 1661). FRIHCP uses a simple handshake (using Configure-Request and Configure-Ack packets only) to enable the header compression algorithms and negotiate their parameters. The algorithms are described in RFC 2507 (IP Header Compression) and RFC 2508 (Compressing IP/UDP/RTP Headers for Low-Speed Serial Links).

Unlike data compression and encryption, there is no separate PPP control protocol for IP header compression. In that case, IP header compression is negotiated over a PPP link as part of the PPP IP Control Protocol (RFC 1332) and IPv6 over PPP (RFC 2472). The FRIHCP, however, is a single control protocol to negotiate header compression for both IPv4 and IPv6.

FRIHCP relies on phases similar to PPP. The order of the phases is as follows:

1. VC Establishment – This phase is controlled by the signaling procedures (PVC (FRF.1.2) or SVC (FRF.4.1)) and is outside the scope of this agreement. The VC is assumed to be established.
2. Header compression negotiation phase – Used to negotiate the parameters to be used for header compression during the data transfer phase.
3. Data Transfer Phase – Transfer of compressed IP headers.

2.1.1 Scope

Frame Relay IP Header Compression is used per VC, end-user to end-user (DTE to DTE). It allows negotiation of header compression protocols and transport of compressed data. This protocol is used exclusively on the Frame Relay user plane, *i.e.*, DLCIs used for user data transfer, not DLCI 0. It is compatible with other Frame Relay Forum protocols – see Section 5.

2.2 Reference Model

The IP header compression procedure is only used between FR DTEs, as shown in Figure 1. The IP header compression procedure is transparent to Frame Relay network(s) between the transmitting and receiving DTEs. When ATM is used in the core, IP header compression will work with FRF.5. Use with service interworking (FRF.8.1) is for further study.

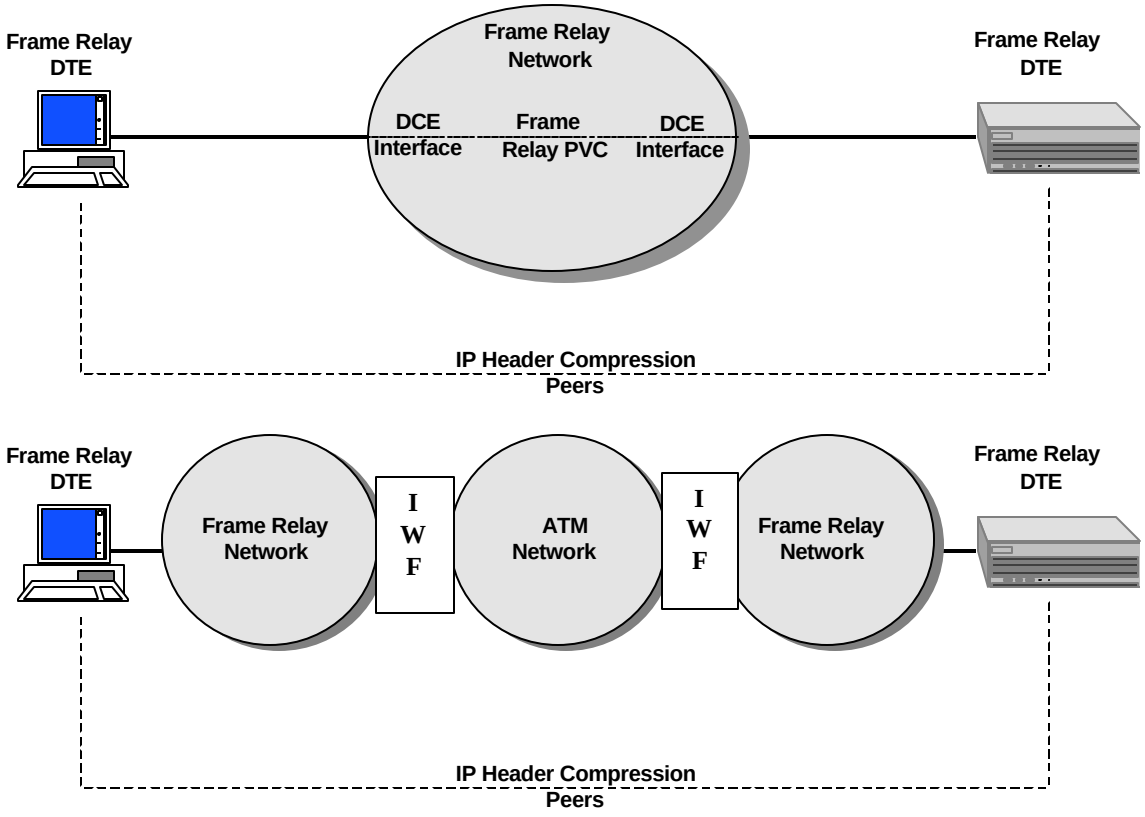


Figure 1 - Reference Diagram

3 ENCAPSULATION

This section describes how Frame Relay IP Header Compression control PDUs and data PDUs are encapsulated using Multiprotocol Encapsulation (FRF.3.2) over a Frame Relay network.

3.1 Data PDU Encapsulation

Standard IPv4 and IPv6 datagrams are encapsulated as shown in Figure 2. These are sent:

- when header compression has not been negotiated,
- during the negotiation procedure, and
- when a compressor decides not to compress a packet header – for example, non-TCP packets will be sent as standard datagrams if only TCP header compression has been negotiated.

Description	Octet
Q.922 Address (2 or 4 octets)	1 … m
Control (0x03)	m+1
NLPID (0xCC or 0x8E)	m+2
IPv4 or IPv6 datagram	m+3 … n
FCS (2 or 4 octets; for 4 octets see FRF.14 Section 13)	n+1 … p

Figure 2 – IP Encapsulation

The IP header compression algorithms (RFCs 2507 and 2508) use 9 packet types in addition to the standard IPv4 and IPv6 packet types. These are used to indicate the different types of compressed packet, with the exception of CONTEXT_STATE, which is used to indicate loss of synchronization. The 9 packet types have been assigned protocol identifiers as shown in Table 1. The format of these packet types is described in the RFCs. Note that the RFCs number the bits in a byte from 0 to 7, where 0 is the most significant bit, whereas Frame Relay specifications number the bits in a byte from 8 to 1, where 8 is the most significant bit.

FULL_HEADER	0x00
COMPRESSED_TCP	0x01
COMPRESSED_TCP_NODELTA	0x02
COMPRESSED_NON_TCP	0x03
COMPRESSED_RTP_8	0x04
COMPRESSED_RTP_16	0x05
COMPRESSED_UDP_8	0x06
COMPRESSED_UDP_16	0x07
CONTEXT_STATE	0x08

Table 1 – Header Compression Packet Identifiers

These packets are encapsulated as shown in Figure 3. Note the use of 0x02 for the control field, which indicates the NLPID is allocated by the Frame Relay Forum.

Compressed packets should only be sent when IP header compression has been successfully negotiated by the FRIHCP.

NLPIDs 0x03-CC (IPv4) or 0x03-8E (IPv6) are used in a Frame Relay VC when IP header compression has not been invoked for an IP packet stream. NLPID 0x02-00 refers to an uncompressed header for a packet stream for which IP header compression was invoked. The definition of IP packet stream appears in RFC 2507.

Description	Octet
Q.922 Address	1 … m
Control (0x02)	m+1
NLPID (Table 1)	m+2
Compressed packet	m+3 … n
FCS (2 or 4 octets; for 4 octets see FRF.14 Section 13)	n+1 … p

Figure 3 – Compressed PDU Encapsulation

3.2 Control PDU Encapsulation

The Frame Relay IP Header Compression control protocol (FRIHCP) messages are encapsulated as shown in Figure 4. These messages are used to negotiate the use of IP header compression.

Description	Octet
Q.922 Address	1 … m
Control (0x02)	m+1
NLPID (0x09)	m+2
Frame Relay IP Header Compression Control PDU	m+3 … n
FCS (2 or 4 octets; for 4 octets see FRF.14 Section 13)	n+1 … p

Figure 4 – Control PDU Encapsulation

4 NEGOTIATION

This section describes the FRIHCP negotiation procedures and PDU formats. The FRIHCP procedures are responsible for enabling and initiating header compression algorithms on both ends of the VC. FRIHCP uses a similar packet exchange mechanism to the PPP Link Control Protocol (RFC 1661).

The use of the header compression facility is negotiated between peer devices. The algorithms are selected independently for each direction of a VC.

To establish compression of IP datagrams sent over a Frame Relay VC, each end of the VC must agree on a set of configuration parameters for the compression. The process of negotiating VC parameters is handled by a Frame Relay IP Header Compression control Protocol (FRIHCP), which is based on the procedures defined for IP Header Compression over PPP (RFC 2509) and for Frame Relay Privacy (FRF.17).

4.1 Specification

Negotiation consists of a simple handshake to enable the IP header compression algorithms and agree the header compression parameters for each direction of the VC. Each direction is negotiated independently, so it is possible to have header compression enabled in one direction only, or have different algorithms and/or parameters enabled in each direction. The IP header compression algorithms are RFC 2507 (IP Header Compression) and RFC 2508 (Compressing IP/UDP/RTP Headers for Low-Speed Serial Links).

4.1.1 Control Frame Formats

Description		Octet
Frame Relay address, control and NLPID information		1 … m
FRIHCP Control Primitive (Note 1) Code		m+1
Identifier		m+2
Length		m+3
(2 octets)		m+4
Configuration Option Type (254)		m+5
Length		m+6
Negotiation Codes	Version	m+7
Parameter Elements		m+8 … n
FCS (2 or 4 octets)		n+1 … p

Note 1: FRIHCP Control Primitive includes octets (m+1) to n

Figure 5 – Control Frame

Field	Description
DLCI, control and NLPID	See Figure 4; octets 1 to 4 (2 octet DLCI) or 1 to 6 (4 octet DLCI).
FRIHCP Control Primitive Code	1 – Config-Req 2 – Config-Ack (values given in decimal)
Identifier	A transaction number to correlate a request with a response. Sent in request and echoed in corresponding response. Use as specified in RFC 1661, Section 5.
Length (2 octets)	Including: Code, Identifier, Length and all Configuration Options
Configuration Options Type	254 (decimal)
Length	Varies depending on number of parameters
Negotiation Codes	4 bits 0000 reply with Response only 0001 reply with Response and initiate Request all other values are reserved This field is not significant in a Config-Ack
Version	Version number of this implementation agreement 4 bits 0001 Version 1.0
Parameter Elements	Zero or more of the parameter elements. See Section 4.1.1.1
FCS	Q.922 Frame Check Sequence

Table 2 – Control Frame

4.1.1.1 Parameter Elements

The Parameter Element ID identifies a parameter element. The length is the length of the whole parameter element including the Parameter Element ID field and the Length field. The Values field lists the individual parameter values of the element. The parameter elements must consist of an integral number of octets. These start at Octet m+8 of the configuration option (Figure 5).

Description	Octet
Parameter Element ID	a
Length	b
Parameter Element Values	C … M

Figure 6 – General Parameter Element Structure

There are two parameter elements:

- IPv4 parameters
- IPv6 parameters.

The presence of the IPv4 and/or IPv6 parameters indicates a desire to enable IPv4 and/or IPv6 header compression. Neither, one or both may be present. If neither is present, header compression will not be enabled. This is used when negotiating header compression in only one direction on the VC.

Both the IPv4 and IPv6 parameters have the same format, based on RFC 2509 (IP Header Compression over PPP). They are distinguished by the Parameter Element ID. The IPv4 ID is 4 (decimal). The IPv6 ID is 6 (decimal).

The parameters are as shown in Figure 7. Octet 1 in Figure 7 corresponds to Octet a in Figure 6. Octet 2 in Figure 7 corresponds to Octet b in Figure 6. Octets 3 to n in Figure 7 correspond to Octets C to M in Figure 6. See Table 3 for descriptions of the parameters.

Description	Octet
IP type ID (4 – IPv4 or 6 – IPv6)	1
Length: >= 12	2
TCP_SPACE	3, 4
NON_TCP_SPACE	5, 6
F_MAX_PERIOD	7, 8
F_MAX_TIME	9, 10
MAX_HEADER	11, 12
Suboptions	13 … n

Figure 7 – Parameters

Field	Description
IP type ID	Octet identifying IPv4 (4) or IPv6 (6) parameter element (values in decimal)
Length	>= 12 (decimal) depending on suboptions
TCP_SPACE	Maximum value of a TCP context identifier (CID) Permitted range: 3 – 255 Suggested value: 15 Special case: 0 implies DO NOT compress TCP headers
NON_TCP_SPACE	Maximum value of a non-TCP context identifier (CID) Permitted range: 3 – 65535 Suggested value: 15 Special case: 0 implies DO NOT compress non-TCP headers
F_MAX_PERIOD	Largest number of compressed non-TCP headers that may be sent without sending a full header Permitted range: 1 – 65535 Suggested value: 256 Special case: 0 implies infinity
F_MAX_TIME	Maximum time interval between full headers Permitted range: 1 – 255 Suggested value: 5 (seconds) Special case: 0 implies infinity
MAX_HEADER	Largest header size that may be compressed Permitted range: 60 – 65535 Suggested value: 168
Suboptions	Zero or more suboptions, comprising type, length and zero or more parameter octets, as defined by the suboption type. RTP-Compression suboption: type = 1, length = 2

Table 3 – Parameters

Like PPP, the parameters indicate the capability of the receiver/de-compressor. The suggested values and permitted ranges are as defined in RFC 2507, with the following important exceptions:

- TCP_SPACE = 0 implies DO NOT compress TCP headers
- NON_TCP_SPACE = 0 implies DO NOT compresses non-TCP headers

The RTP-Compression suboption from RFC 2509 is used if IP/UDP/RTP compression is to be enabled. It must not be present if NON_TCP_SPACE is 0. Any future suboptions, which get defined for use with this header compression protocol over PPP, are allowed.

The IPv4 and/or IPv6 parameters are included in the Configure-Request if the implementation is capable of receiving compressed IPv4 and/or IPv6 headers. A non-zero value for TCP_SPACE and/or NON_TCP_SPACE indicates compressed TCP and/or non-TCP packets can be handled. The presence of the RTP-Compression suboption indicates compressed IP/UDP/RTP packets can be handled.

The IPv4 and/or IPv6 parameters are included in the Configure-Ack if they were present in the received Configure-Request and the sender of the Ack agrees to send compressed IPv4 and/or IPv6 packets. The values of the parameters sent in the Configure-Ack must not be larger than the values received in the corresponding Configure-Request. The Configure-Ack must not contain any suboptions that were not present in the corresponding Configure-Request. If the IPv4 and/or IPv6 parameters were present in the Configure-Request but omitted from the Configure-Ack, then IPv4 and/or IPv6 header compression will not be performed in the direction from the sender of the Ack to the sender of the Request.

To help conserve memory on the receiver, the TCP_SPACE, NON_TCP_SPACE and/or MAX_HEADER values may be reduced in the Configure-Ack if the sender of the Ack intends to use lower values. The TCP_SPACE or NON_TCP_SPACE value may be reduced to zero in the Configure-Ack if the sender of the Ack does not intend sending compressed TCP or non-TCP headers. Any suboptions in the Configure-Request may be omitted from the Ack if the sender of the Ack does not wish to enable the suboption.

4.1.1.1.1 Example

The following figure shows a complete Configure-Request PDU for negotiating IPv4 TCP and RTP header compression with the suggested parameters. The example assumes a 2 octet DLCI.

Description	Octet
Frame Relay address, control and NLPID information (Figure 4)	1 … 4
Code: Config-Req (1)	5
Identifier	6
Length (21)	7, 8
Type (254)	9
Length (17)	10
Negotiation Code (0001), Version (0001)	11
Parameter ID: IPv4 (4)	12
Length (14)	13
TCP_SPACE (15)	14, 15
NON_TCP_SPACE (15)	16, 17
F_MAX_PERIOD (256)	18, 19
F_MAX_TIME (5)	20, 21
MAX_HEADER (168)	22, 23
Type: RTP-Compression (1)	24
Length: 2	25

Figure 8 – IPv4 Configure-Request

4.1.2 Control Procedures

FRIHCP provides a simple negotiation protocol to enable header compression and agree parameter values for each direction of the VC. Once FRIHCP is successfully enabled, IP data transfer to/from the peer end system may be header compressed. To disable FRIHCP, an implementation may force the virtual connection to the inactive state, or send a request and not send a response.

FRIHCP consists of three phases: Disabled, Initialization, and Operational. The Disabled phase is entered upon power-up or when a Frame Relay virtual connection is released. The Initialization phase is entered upon Frame Relay virtual connection establishment and when FRIHCP is enabled. The Operational phase is entered upon the successful completion of the Initialization phase. Unsuccessful completion of the Initialization phase causes FRIHCP to enter the Disabled Phase. Compressed data PDUs (Table 1) are transferred only when in the Operational phase. FRIHCP control PDUs may be transferred in any phase.

4.1.2.1 States

The FRIHCP states, which may exist on either side of the Frame Relay connection, are:

Disabled (D): (initial state)

FRIHCP does not exist.

Request Initiated (I_1)

A Configure-Request message has been sent to the peer. Awaiting Configure-Ack to own request and peer Configure-Request.

Request Received (I_3)

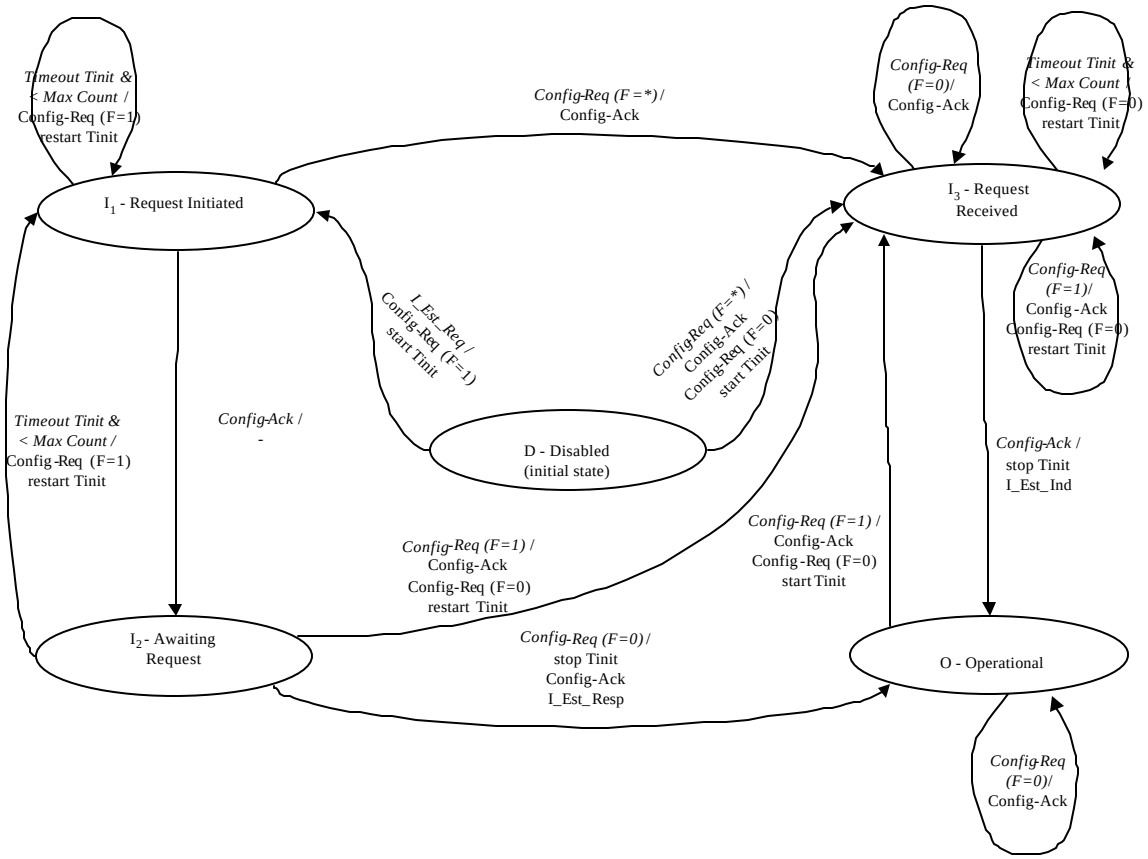
A Configure-Request message has been received from the peer. A Configure-Ack to the peer request message, and a Configure-Request are sent to the peer. Awaiting Configure-Ack to own request.

Awaiting Request (I_2)

Received Configure-Ack to own request and awaiting the peer Configure-Request.

Operational (O)

Negotiation completed.



Note: F indicates the Negotiation Codes defined in Table 2 (* means either 0 or 1).

I_Est_Req, I_Est_Resp and I_Est_Ind are internal primitives; not exchanged between FRIPHC peers.

Not all transitions to the *Disabled (D)* state are shown. Consult the text for the complete description.

Figure 9 – State Diagram

4.1.2.2 Initialization Request

The Initialization will start when a Frame Relay virtual connection to a peer is established and FR IP Header Compression function is administratively enabled (by the user – I_Est_Req). A signal for PVC establishment is obtained via link management Procedures (*e.g.*, Q.933 Annex A) when there is a transition from a PVC status of "inactive" to "active" and/or the presence of both "active" and "new" for a PVC. A signal for SVC establishment is obtained via Q.933 call control procedures when a call transitions to the Active state.

Frame Relay IP Header Compression protocol negotiation procedures are initiated, by sending a Configure-Request message with negotiation codes set to "reply with response and initiate request" to the peer, start a handshake completion timer, and enter *Request Initiated (I_1)* state.

Upon receiving a Configure-Ack message, the entity shall enter *Awaiting Request (I_2)* state. When a Configure-Request is received from the peer, the following procedure shall apply:

1. For calls in *Request Initiated (I_1)* state, send a Configure-Ack message, and enter *Request Received (I_3)* State.
2. For calls in *Awaiting Request (I_2)* state, the action taken depends on Configure-Request message negotiation code.

- If the negotiation code is set to "reply with response only", send a Configure-Ack message, stop handshake completion timer, send a I_Est_Resp internal primitive to indicate that the negotiation is complete, and enter *Operational (O)* state.
- If the negotiation code is set to "reply with response and initiate request", send a Configure-Ack message and a Configure-Request message with negotiation codes set to "reply with response" to the peer, restart a handshake completion timer, and enter *Request Received (I_3)* state.

If the handshake completion timer expires before the handshake procedure is completed and the number of retries are less than Max Count, the following procedure shall apply:

1. For calls in *Request Initiated (I_1)* state, send a Configure-Request message with negotiation codes set to "reply with response and initiate request" to the peer, and restart a handshake completion timer.
2. For calls in *Awaiting Request (I_2)* state, send a Configure-Request message with negotiation codes set to "reply with response and initiate request" to the peer, restart a handshake completion timer, and enter *Request Initiated (I_1)* state.

4.1.2.3 Receipt of a Configuration Request

Upon receipt of a Configure-Request from the peer in *Disabled (D)* state, send a Configure-Ack message and a Configure-Request message with negotiation codes set to "reply with response" to the peer, start a handshake completion timer, and enter *Request Received (I_3)* state.

Upon receiving a Configure-Ack message in the *Request Received (I_3)* state, Stop handshake completion timer, send a I_Est_Ind internal primitive to indicate that the negotiation is complete, and enter *Operational (O)* state.

When a Configure-Request message is received from the peer, for calls in *Request Received (I_3)* state, the action taken depends on the message negotiation code:

- If the negotiation code is set to "reply with response only", send a Configure-Ack message.
- If the negotiation code is set to "reply with response and initiate request", send a Configure-Ack message and a Configure-Request message with negotiation codes set to "reply with response" to the peer, and restart a handshake completion timer.

If the handshake completion timer expires before the handshake procedure is completed and the number of retries are less than Max Count, send a Configure-Request message with negotiation codes set to "reply with response" to the peer, and restart a handshake completion timer.

4.1.2.4 Operational Phase

When a Configure-Request message is received from the peer, for calls in the *Operational (O)* state, the action taken depends on the message negotiation code:

- If the negotiation code is set to "reply with response only", send a Configure-Ack message.
- If the negotiation code is set to "reply with response and initiate request", send a Configure-Ack message and a Configure-Request message with negotiation codes set to "reply with response" to the peer, restart a handshake completion timer, and enter *Request Received (I_3)* state.

4.1.2.5 Disabled Phase

The Disabled phase shall be entered when a Frame Relay virtual connection to a peer is released. A Frame Relay virtual connection is released when a signal for PVC inactive is obtained via link management Procedures (*e.g.*, Q.933 Annex A), or a signal for SVC release is obtained via Q.933 call control procedures.

If Max Count is exceeded on a handshake completion timer expiry, return to the *Disabled (D)* state and notify the management entity.

4.1.2.6 Timer and Counter values for the state machine

The recommended default value for the handshake timer is 3 seconds, and the suggested range is from 1-10 seconds. The recommended default value for Max Count is 3 and the suggested range is from 1-10.

4.1.2.7 Error Handling

As per RFC 1661, invalid packets are silently discarded without affecting the state machine. The implementation should provide the capability of logging the error, including the contents of the silently discarded packet, and should record the event in a statistics counter.

4.1.3 Data Transfer Procedures

Once the negotiation between the two header compression peers is completed and both peers are in the *Operational (O)* state, IP headers are compressed using the procedures defined in RFCs 2507 and 2508.

5 INTERACTION WITH OTHER PROTOCOLS

The following are examples of frame formats used with FRIPHC data transfer and other Frame Relay Forum implementation agreements. In general, the order of protocol application is header compress, encrypt, and fragment.

5.1 Data Compression over Frame Relay (FRF.9)

It is possible to negotiate both header compression and payload compression on the same VC. However, it is not possible to apply both header compression and payload compression to the same packet. This is because FRF.9 can only compress packets encapsulated with control 0x03 – header compressed packets are encapsulated with control 0x02. Only those packets that do not get header compressed can be payload compressed.

5.2 Privacy (FRF.17)

The following example shows a header compressed IP packet that is encrypted using FRF.17 Mode 1. The compressed frame starting with the control field, not including the FCS, is encrypted and becomes the payload of the FRF.17 frame. This example assumes a 2 octet DLCI.

Description			Octet
Q.922 Address			1, 2
Control (UI: 0x03)			3
NLPID (0xB3)			4
FRPP Header			
ext 1	Spare	C/D 0	5
Sequence number			6
Ciphertext Identifier (Note 1)			7
FRIPHC Information (Note 1)	**Control (0x02)**		**8**
	NLPID (Table 1**)**		**9**
	Compressed packet		**10 … m**
Self Describing Padding (Note 1)			m+1 … n
LCB			n+1
FCS (2 or 4 octets)			n+2 … p

Note 1: This field is encrypted.

Figure 10 – Encrypted FRFIPHC frame

5.3 Fragmentation (FRF.11.1 & FRF.12)

The following example shows a header compressed IP packet that is fragmented using FRF.12 end-to-end fragmentation. A header compressed IP packet is fragmented the same as any other frame. The compressed packet, starting with the control field and not including the FCS, is fragmented and becomes the payload fragments of the FRF.12 frames. This is the same method used for FRF.11.1 Annex J. FRF.12 UNI/NNI fragmentation is done as specified in FRF.12.

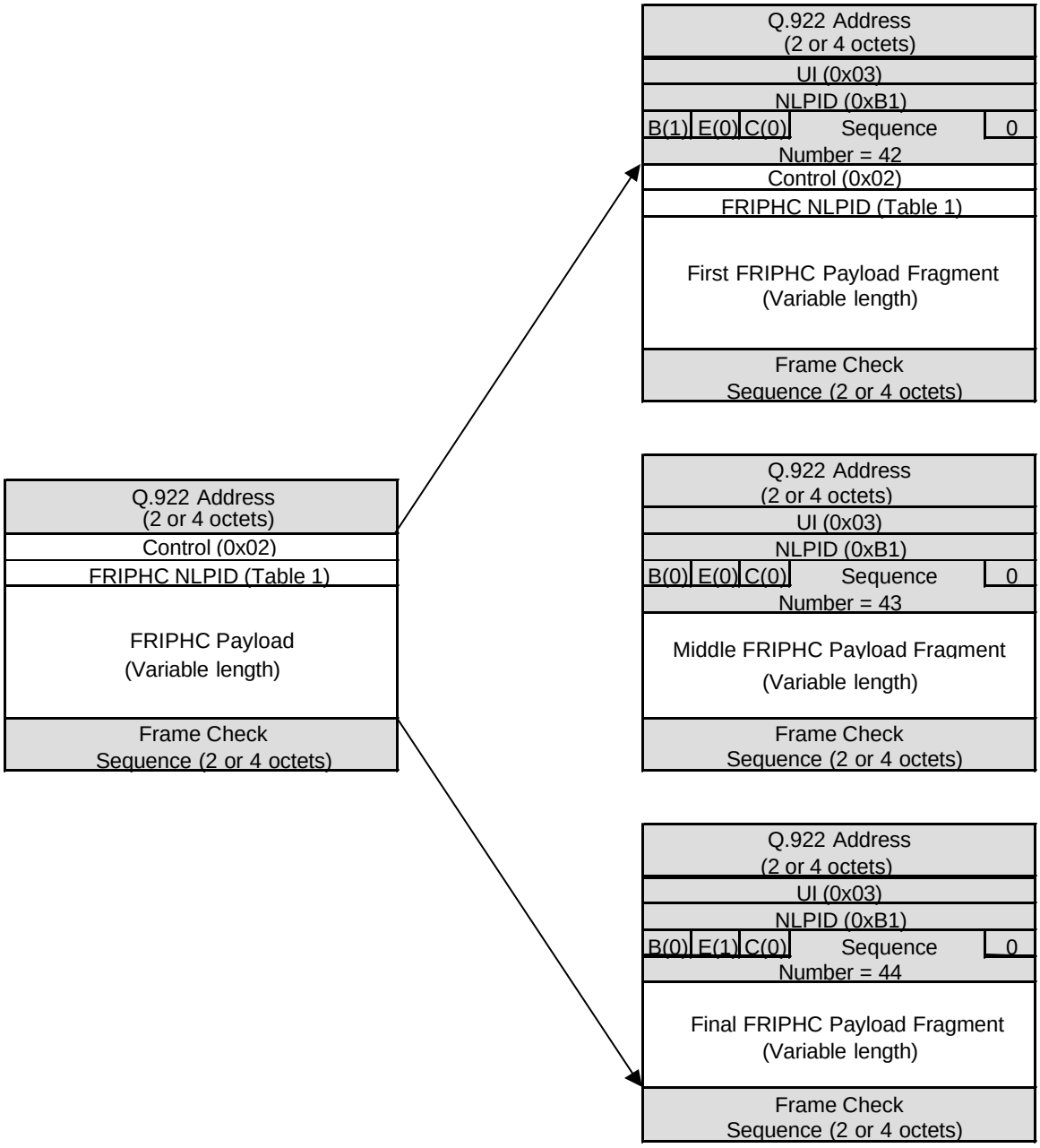


Figure 11 – Compressed frame fragmented with FRF.12 (end-to-end) and FRF.11.1 Annex J

For FRF.11.1, without using Annex J, a similar method would be used for data sent in the voice frame structure. The data would first be compressed, and then fragmented according to the procedures in FRF.11.1. It would then be encapsulated in the voice frame structure described in FRF.11.1.