Multilink Frame Relay UNI/NNI Implementation Agreement

FRF.16

Frame Relay Forum Technical Committee
August 1999

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 take 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:

Mike Sheehan
Ascend Communications Corporation

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

Table of Contents

List of Tables

List of Figures

Revision History

Version	Change	Date
FRF.MFR-UN	Document Approved	TBD

1   Introduction

1.1   Purpose

Multilink Frame Relay (MFR) for the User-to-Network Interface (UNI) and the Network-to-Network Interface (NNI) provides physical interface emulation for frame relay devices. The emulated physical interface consists of one or more physical links, called "bundle links", aggregated together into a single "bundle" of bandwidth. This service provides a frame-based inverse multiplexing function, sometimes referred to as an "IMUX".

The bundle provides the same order-preserving service as a physical layer for frames sent on a data link connection. In addition, the bundle provides support for all Frame Relay services based on UNI and NNI standards.

The agreements herein were reached in the Frame Relay Forum, and are based on the relevant Frame Relay and Internetworking standards referenced in Section 1.6. They document agreements reached among vendors and suppliers of Frame Relay network products and services regarding Frame Relay.

This document may be submitted to different bodies involved in ratification of implementation agreements and conformance testing to facilitate multi-vendor interoperability, and to different standards bodies for inclusion in international standards.

1.2   Multilink Frame Relay Applications

Multilink Frame Relay solves the following problems on user-to-network and network-to-network interfaces in a frame relay network:

- Lack of required bandwidth availability due to facility constraints (e.g. no E3/T3 service in a geographical region) or service offering restrictions (e.g. no fractional E1/T1 service),

- The physical interface as an inflexible pool of bandwidth, and

- The physical interface as a single point of failure on the frame relay interface.

By combining multiple physical interfaces into a single bundle, a network operator can design a frame relay interface supporting more bandwidth than is available from any single physical interface. Further, this agreement provides techniques that, if applied by equipment vendors, support the dynamic addition and removal of physical interfaces to change the total bandwidth available on the interface. Finally, resilience is provided when multiple physical interfaces are provisioned on a single bundle so that when some of the physical interfaces fail, the bundle continues to support the frame relay service.

1.3   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 implementor.

1.4   Terminology

Physical Link --	A single physical interface that interconnects two devices in a frame relay network (e.g. DS1, DS0, Bearer channel, refer to FRF.14).
Bundle —	A grouping of one or more physical links using the formats and procedures of multilink frame relay. The bundle operates as a logical interface function that emulates a single physical interface to the Q.922 data link layer.
Bundle Link—	A MFR sub-component that controls operation of one of the bundle's physical links.
MFR Control Frame —	A frame used to exchange MFR Link Control information.
MFR Information Frame —	A frame used to transport Q.922 data frames (i.e. frame relay signaling and data frames).

1.5   Acronym List

DCE	Data Circuit-terminating Equipment
DLCI	Data Link Connection Identifier
DTE	Data Terminal Equipment
FR	Frame Relay
HDLC	High Level Data Link Control
IA	Implementation Agreement
IMUX	Inverse Multiplexer
MDL	Communication between layer management entity and data link layer
MFR	Multilink Frame Relay
MTU	Maximum Transmission Unit
OUI	Organizationally Unique Identifier
NNI	Network-to-Network Interface
PVC	Permanent Virtual Connection
SVC	Switched Virtual Connection
UNI	User-to-Network Interface

1.6   Relevant Standards

The following is a list of standards on which these implementation agreements are based:

[1]   FRF.3.1   R. Cherukuri (ed.), Multiprotocol Encapsulation Implementation Agreement, June 22, 1995.

[2]   RFC 2427   C. Brown, A. Malis, Multiprotocol Interconnect over Frame Relay, September 1998

[3]   FRF.12   Frame Relay PVC Fragmentation Implementation Agreement. December 1997.

[4]   FRF1.1   User-to-Network Implementation Agreement, January 18, 1996.

[5]   FRF.14   K. Rehbehn, Physical Layer Implementation Agreement.

[6]   RFC 1990,   K. Sklower, B. Lloyd, G. McGregor, D. Carr, T. Coradetti, The PPP Multilink Protocol (MP), August 1996.

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

[8]   Q.921   ITU-T. *Recommendation Q.921, ISDN USER-NETWORK INTERFACE-DATA LINK LAYER SPECIFICATION*

[9]   Q.933   ITU-T. *Recommendation Q.933, ISDN Signaling Specification for Frame Mode Bearer Services.*

[10]   RFC 1661   W. Simpson, The Point-to-Point Protocol (PPP), July 1994

2   Reference Model

This implementation agreement provides two types of interface-based multilink support for frame relay: the User-to-Network interface (UNI) and the Network-to-Network interface (NNI).   A MFR interface consists of one or more bundle links bound together to form a bundle. The bundle operates as a logical interface function that emulates a single physical interface.

Each type of bundle is described below. The two types of bundles use identical message formats and protocol procedures.

2.1 UNI

The UNI-based MFR interface connects data terminal equipment (DTE) to the network data circuit-terminating equipment (DCE). Figure 1 illustrates the structure of this interface.
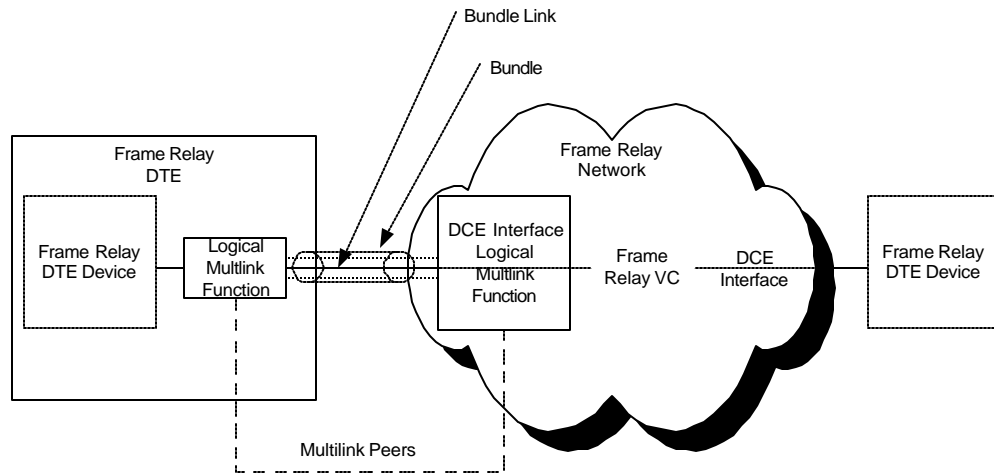
Figure 1 UNI Reference Model

2.2 NNI

Figure 2 illustrates the structure of the NNI-based Multilink Frame Relay interface. One or more bundle links are bound together to form a bundle between adjacent DCE devices.
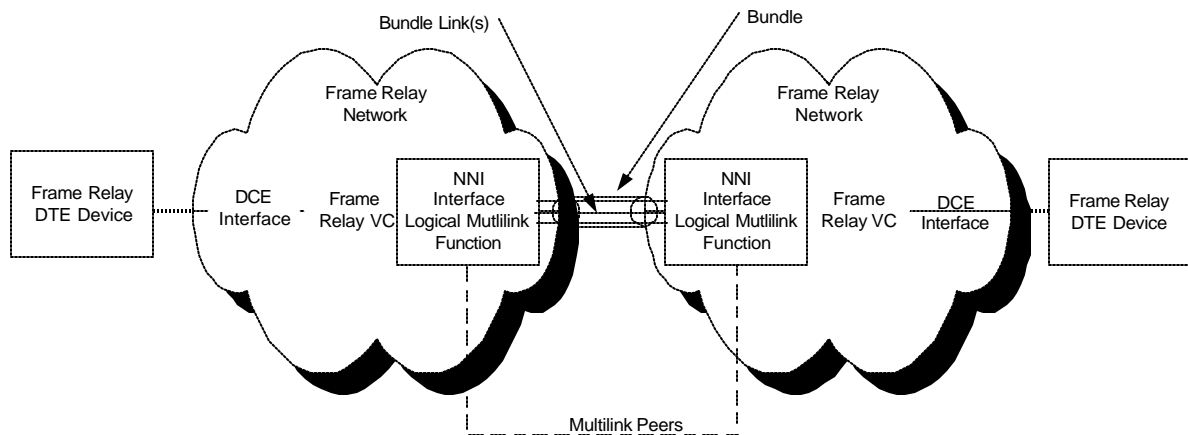
Figure 2 NNI Reference Model

2.3   Protocol Stack

The bundle operates in conjunction with the Q.922 data link layer. It emulates the functions of the physical layer.
Figure 3 illustrates the relationship between the bundle and the other elements of the protocol stack. The Data Link Layer multiplexes frames from the different data link connections, including the data link connection assigned to signaling for the C-Plane. Frames are exchanged with the bundle Layer which emulates a physical interface for the transport of frames. The bundle Layer assigns frames to one or more bundle links for transport to the peer over a real instance of a Physical Layer.

The bundle Layer may dispatch frames over any bundle link. The receiving bundle Layer must ensure that a data link connection's frame order is preserved. The bundle Layer may also implement fragmentation procedures to limit the maximum size of a frame transmitted over a bundle link.
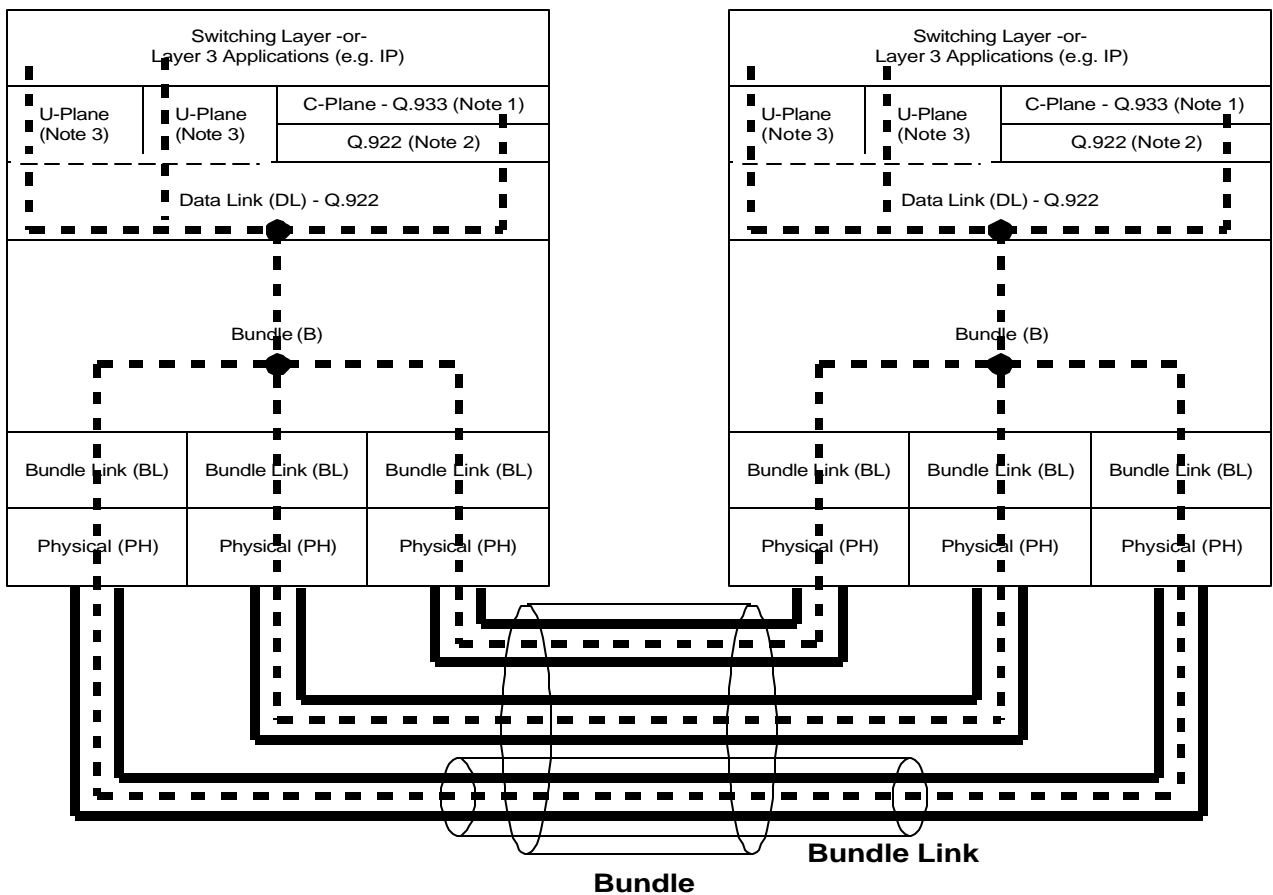
Note 1: C-Plane operation as described in Q.933 [9] and FRF.4 [11]
Note 2: Mutiple frame acknowledged information transfer mode as described in Q.922 [7]
Note 3: Core aspects for use with frame relaying bearer service as described in Q.922, Annex A [7]

Figure 3
Protocol Stack for MFR

2.4   Inter-layer Communication

Primitives are used as an abstract description of the control interface between a bundle link, other layers, and layer management. Primitives do not specify or constrain implementations. Table 1 illustrates the primitives defined in this agreement.

The bundle presents a physical layer interface to the data link layer. The MFR bundle link utilizes the actual physical layer. Both instances use the same physical layer primitives found in Q.921[8].

The primitives unique to the bundle are:

MB_ADD_LINK	The MB_ADD_LINK request primitive is used to request the addition of a bundle link to the active operation of a bundle.
MB_REMOVE_LINK	The MB_REMOVE_LINK primitive is used to request the transition of a bundle link from active operation to an idle state. Once issued, the endpoint must not attempt to transmit additional frames on the bundle link. However, the endpoint should continue to accept frames until the MB_REMOVE_LINK response primitive is issued following receipt of the acknowledgement from the peer endpoint.
MB_ERROR	The MB_ERROR primitive reports error conditions detected during bundle operation
BL_ACTIVATE	The BL_ACTIVATE primitive controls the addition of an individual bundle link to a bundle. The confirmation is returned when the bundle link is ready to transmit and receive frames.
BL_DEACTIVATE	THE BL_DEACTIVATE primitive controls the removal an individual bundle link. After the bundle requests removal of a bundle link, the bundle must not transmit any frames on that bundle link. However, the bundle may continue to receive frames from the bundle link until the BL_DEACTIVATE confirmation is received. BL_DEACTIVATE indication is issued following detection of error conditions on the bundle link.
BL_DATA	The BL_DATA primitive forwards a single fragment to and from an individual bundle link.
PH_ACTIVATE	Refer to Q.921 [8]. Note that the bundle Layer emulates the physical layer to the data link layer. The bundle supports a set of physical layer primitives to provide an emulated physical layer for the data link layer. The individual bundle links also utilize the physical layer primitives to interact with a real instance of a physical link.
PH_DEACTIVATE	Refer to Q.921 [8].
PH_DATA	Refer to Q.921 [8].

Generic Name	Type			Parameter Data Contents
	Request	Ind	Confirm	
MB_ADD_LINK	X	X		bundle Identification, Link Identification
MB_REMOVE_LINK	X	X		Link Identification
MB_ERROR	-	X		Reason for error
BL_ACTIVATE	X		X	bundle Identification, Link Identification
BL_DEACTIVATE	X	X	X	Cause, Diagnostic Field
BL_DATA	X	X		Fragmentation Frame
PH_ACTIVATE	X	X		
PH_DEACTIVATE		X		
PH_DATA	X	X		Data Link Frame

Table 1
Primitives

Figure 4 illustrates the relationship between the protocol layers.
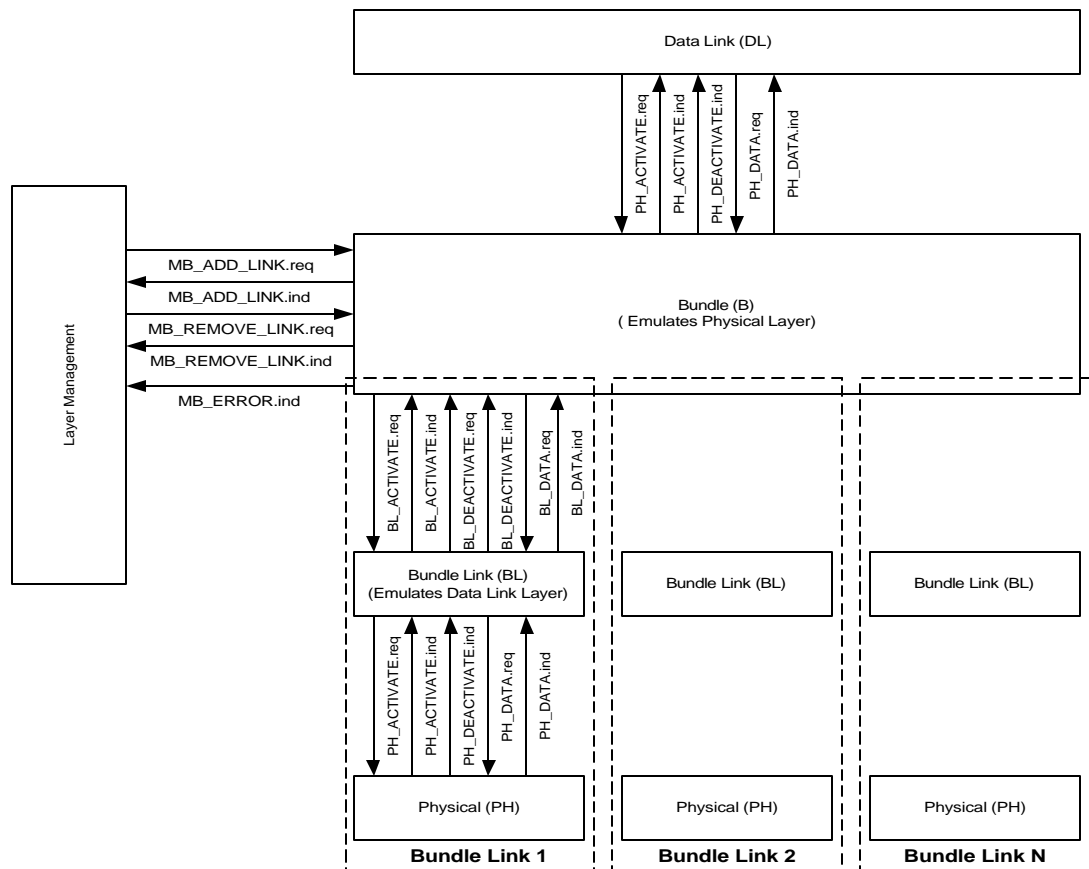
Figure 4 Relationship Between Layers

2.5 Q.933 Annex A Status Procedures

The Q.933 Annex A Status Procedure, operating on a MFR interface, transmits frames (e.g. FULL STATUS REQUEST message) using the PH_DATA request primitive. Incoming messages are received via the PH_DATA indication primitive.

The bundle links of the bundle do not perform special procedures in support of the Q.933 Annex A Status Procedure. All frames sent on the signaling DLCI are scheduled over the bundle link in the same way as user plane data is scheduled.

3 Formats

3.1 MFR Fragmentation Frame Format

Frame relay frames transmitted on the bundle are encapsulated in an MFR fragmentation frame, as shown in Figure 5. Fragment structure and fragmentation procedures are described in [FRF.12]. The UNI and NNI interface fragmentation techniques described in FRF.12[3]apply. The (C)ontrol Bit described in Section 5.1 of FRF.12[3] must be set to zero for all MFR fragmentation frames.

<div align="center">Bits</div>

8	7	6	5	4	3	2	1	Octets
B	E	0	Sequence Number (msb)				1	1
Sequence Number (lsb)								2
DLCI (msb)					C/R		0	3
DLCI (lsb)			FECN	BECN	DE		1	4
Fragment Payload								5

<div align="center">

Figure 5
MFR Fragmentation Frame

</div>

3.2   Link Integrity Protocol Control Message Format

MFR Link Integrity Protocol control messages are transmitted in MFR frames that have the (C)ontrol Bit set to one. All messages are sent in a single fragment (B=1 and E=1). The message consists of a Message Type Field and multiple variable length information fields.

The format of the MFR Link Integrity Protocol message is shown in Figure 6.

<div align="center">Bits</div>

8	7	6	5	4	3	2	1	Octets
1 B	1 E	1 C	0	0 Spare	0	0	1	1
0	0	0	0	0 Spare	0	0	0	2 (Note 1)
Message Type									3 (Note 2)
Information Field 1									4
Information Field N									N

NOTES:
1. Spare bits should not be set. A message with the spare bits set shall not be rejected.

2. Message Type values are:
   1   ADD_LINK
   2   ADD_LINK_ACK
   3   ADD_LINK_REJ
   4   HELLO
   5   HELLO_ACK
   6   REMOVE_LINK
   7   REMOVE_LINK_ACK

Figure 6
MFR Link Integrity Protocol Message Format

The format of an information field is shown in Figure 7. Each link integrity protocol message consists of one or more information fields. The message's information fields must be in ascending order by type.

<div align="center">

Bits

8	7	6	5	4	3	2	1	Octets
Type								1 (Note 1)
Length								2 (Note 2)
Data								3 - N

</div>

NOTES:
1. The following information field type encodings are used:
    1   Magic number
    2   Bundle identification
    3   Link identification
    4   Timestamp Information
    5   Vendor extension
    6   Cause

2. Length includes the Type, Length, and Data sub-fields.

Figure 7
MFR Link Integrity Protocol Information Field Format

3.3   Control Messages

3.3.1   ADD_LINK Message

The ADD_LINK message notifies the peer endpoint that the local endpoint supports frame processing. The message includes information required to verify bundle membership and detect loopbacks. Both ends of a bundle link generate this message when a bundle link endpoint is ready to become operational.

The ADD_LINK message must contain the Bundle Identification Information Field (Section 3.4.1), the Link Identification Information Field (Section 3.4.2), and the Magic Number Information Fields (Section 3.4.3).

The ADD_LINK message may contain the optional Vendor Extension Information Field (Section 3.4.5) and Timestamp Information Field (Section 3.4.4).

3.3.2   ADD_LINK_ACK Message

The ADD_LINK_ACK message notifies the peer endpoint that the local endpoint has received a valid ADD_LINK message.

The ADD_LINK_ACK message must contain the Bundle Identification Information Field (Section 3.4.1), the Link Identification Information Field (Section 3.4.2), and the Magic Number Information Field (Section 3.4.3). The values contained in the Bundle Identification Information Field and Link Identification Information Field reflect the identity of the bundle link endpoint transmitting the ADD_LINK_ACK message.

The ADD_LINK_ACK message must contain the timestamp received in the last received ADD_LINK message, if the Timestamp Information Field was present in the last received ADD_LINK message.

The ADD_LINK_ACK message may contain the optional Vendor Extension Information Field (Section 3.4.5) with the same Organizationally Unique Identifier (OUI) as was present in the last received ADD_LINK message.

3.3.3   ADD_LINK_REJ Message

The ADD_LINK_REJ message notifies the peer endpoint that the local endpoint has received an invalid ADD_LINK message.

The ADD_LINK_REJ message must contain the Bundle Identification Information Field (Section 3.4.1), the Link Identification Field (Section 3.4.2), the Magic Number Information Field (Section 3.4.3), and the Cause Information Field (Section 3.4.6). Note that the values contained in the Bundle Identification Information and Link Identification Information Fields reflect the identity of the bundle link endpoint transmitting the ADD_LINK_REJ message and should be communicated to the layer management function.

The ADD_LINK_REJ message may contain the optional Vendor Extension Information Field (Section 3.4.5). If included, the Vendor Extension Information Field must contain the same Organizationally Unique Identifier (OUI) as was present in the last received ADD_LINK message.

3.3.4   HELLO Message

The HELLO message notifies the peer endpoint that the local endpoint remains in the state *up*. Both ends of a bundle link generate this message on a periodic basis.

The HELLO message must contain the Magic Number Information Field (Section 3.4.3).

The HELLO message may contain the optional Vendor Extension Information Field (Section 3.4.5). The Vendor Extension Information Field must have an OUI that is the same as the OUI exchanged between the bundle link endpoints in the ADD_LINK and ADD_LINK_ACK messages.

The HELLO message may contain the optional Timestamp Information Fields (Section 3.4.4).

3.3.5   HELLO_ACK Message

The HELLO_ACK message notifies the peer that the local endpoint has received a valid HELLO message.

The HELLO_ACK message must include the Magic Number Information Field (Section 3.4.3).

The HELLO_ACK message must contain the timestamp received in the last received HELLO message, if the Timestamp Information Field was present in the last received HELLO message.

The HELLO_ACK may contain the optional Vendor Extension Information Field (Section 3.4.5).

If included, the Vendor Extension Information Field must contain the same Organizationally Unique Identifier (OUI) as was present in the last received HELLO message.

3.3.6   REMOVE_LINK Message

The REMOVE_LINK message notifies the peer that the local end layer management function is removing the bundle link from bundle operation.

The REMOVE_LINK message must include the Magic Number (Section 3.4.3) and Cause Information Fields (Section 3.4.6).

The REMOVE_LINK message may contain the optional Vendor Extension Information Field (Section 3.4.5). If included, the Vendor Extension Information Field must contain the same Organizationally Unique Identifier (OUI) as was present in the last received REMOVE_LINK message.

3.3.7   REMOVE_LINK_ACK Message

The REMOVE_LINK_ACK message notifies the peer that the local end has received a REMOVE_LINK message.

The REMOVE_LINK_ACK message must include the Magic Number Information Field (Section 3.4.3)

The REMOVE_LINK_ACK message may contain the optional Vendor Extension Information Field (Section 3.4.5). If included, the Vendor Extension Information Field must contain the same Organizationally Unique Identifier (OUI) as was present in the last received REMOVE_LINK_ACK message.

3.4   Information Fields

3.4.1   Bundle Identification Information Field

The Bundle Identification Information Field provides information used to associate a local endpoint and a remote endpoint with a specific bundle. Assignment of bundle links to specific bundles is performed via the layer management function.

The value used for bundle identification has significance for both local and remote endpoints. The local endpoint assigns the bundle identification to group bundles while the remote end use the bundle identification to verify that the configuration of the bundle identification is consistent with other bundle links in the same bundle. The value may or may not be the same as the value created by the other endpoint on the link.

Figure 8 shows the Bundle Identification Information Field.

				Bits				
8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	0	1	1
Bundle Identification Information Field								
Length								2 (Note 1)
Bundle Identification								3-N (Note 2)

NOTES:
1.   Bundle identification is limited to a length of 50 octets.

2.   Bundle identification must be formatted as a null terminated text string consisting of the ASCII characters A-Z, a-z, 0-9, and the printable characters `~!@#$%^&*()-_=+[]{}\|;:'",.<>/?.

Figure 8
Bundle Identification Information Field

3.4.2 Link Identification Information Field

The Link Identification Information Field provides information used to report the identity of a bundle link when error conditions arise at an endpoint. One example is the inadvertent assignment of a link to the wrong bundle. The endpoint that detects the mismatch uses the information in the received Link Identification Information Field (e.g., a port identifier) to create a report that is meaningful to the layer management function.

The value used for link identification has significance for both local and remote endpoints. The local endpoint assigns the link identification to signify that this link is part of a certain bundle, while the remote end uses the link identification to indicate an error if the link in question has an error associated with it, i.e. misconfiguration. The value may or may not be the same as the value created by the other endpoint on the link.

Assignment of the bundle link identification value is performed via the layer management function.

Figure 9 shows the Link Identification Information Field.

						Bits		
8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	1	0	1
Link Identification Information Field								
Length								2 (Note 1)
Link Identification								3-N (Note 2)

NOTES:
1. Link identification is limited to a length of 50 octets.

2. Link identification must be formatted as a null terminated text string consisting of the ASCII characters A-Z, a-z, 0-9, and the printable characters `~!@#$%^&*()-_=+[]{}\|;:'",.<>/?.

Figure 9
Link Identification Information Field

3.4.3 Magic Number Information Field

The Magic Number Information Field provides information required for looped-back bundle link detection.

Figure 10 shows the Magic Number Information Field.

				Bits				
8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	0	1	1	1
Magic Number Information Field								
Length								2
Magic Number (MSB)								3
Magic Number								4
Magic Number								5
Magic Number (LSB)								6

Figure 10
Magic Number Information Field

3.4.4 Timestamp Information Field

The Timestamp Information Field may be included in an ADD_LINK or HELLO message to encode a local time value that represents the time of transmission. A peer that receives this information field in an ADD_LINK or HELLO message must include the information field in the ADD_LINK_ACK or HELLO_ACK message respectively.

The contents of the Timestamp Information Field are transmitted unchanged back to the originating endpoint. The Timestamp represents a local clock value at the time of ADD_LINK or HELLO message transmission. The value is echoed in the ADD_LINK_ACK or HELLO_ACK message. The use of the timestamp field is explained in Section 4.2.2.4. Granularity and interpretation of the Timestamp Information Field is implementation specific.

Figure 11 shows the Timestamp Information Field.

				Bits				
8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	0	1	1
Timestamp Information Field								
Length								2 (Note 1)
Transmit Timestamp (MSB)								3
Transmit Timestamp								4
Transmit Timestamp (LSB)								N

NOTES:
1. The maximum length is 14 octets. Format is implementation specific.

Figure 11
Timestamp Information Field

3.4.5   Vendor Extension Information Field

The Vendor Extension Information Field extends bundle link procedures to meet vendor-specific requirements. The content of the sub-code and vendor-supplied values sub-fields are not standardized.

Figure 12 shows the Vendor Extension Information Field.

				Bits				
8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	1	0	1
Vendor Extension Information Field								
Length								2
Organizationally Unique Identifier (MSB)								3 (Note 1)
Organizationally Unique Identifier								4
Organizationally Unique Identifier (LSB)								5
Sub-Code								6 (Note 2)
Vendor Supplied Values								7-N

NOTES:
1.   The IEEE assigns the OUI. The three-octet Organizationally Unique Identifier (OUI) identifies the organization that defines the format of this vendor supplied information field.

2.   Sub-code is not standardized. Sub-code values are OUI-specific.

Figure 12
Vendor Extension Information Field

3.4.6 Cause Information Field

The Cause Information Field informs the peer endpoint of the reason for transmission of the ADD_LINK_REJ or REMOVE_LINK message.

Figure 13 shows the Cause Information Field.

Bits								
8	7	6	5	4	3	2	1	Octets
0	0	0	0	0	1	1	1	1
Cause Information Field								
Length								2 (Note 1)
Cause								3 (Note 2)
Diagnostic Information								4 - N (Note 3)

NOTES:
1. The maximum length is 53 octets.

2. Valid cause values are described in Section 4.3.9.

3. Diagnostic values are described in Section 4.3.9. This variable length field may be empty. When empty, no octets are included and the Information Field consists of the tag, length, and cause fields.

Figure 13
Cause Information Field

4 Procedures

4.1 Overview

The bundle contains and controls one or more bundle links supporting the transfer of MFR frames. Bundle procedures provide for the following activities:

- addition of bundle links to bundle operation,

- graceful removal of bundle links from bundle operation,

- interfacing with layer management functions,

- accepting frames from the Q.922 data link layer for transmission on the bundle interface,

- operating frame fragmentation procedures,

- scheduling frames for transmission on individual bundle links, and

- reassembling received frame fragments for forwarding to the Q.922 data link layer.

Each of the bundle's bundle links operates a Link Integrity Protocol that provides the following features:

- confirmation of bundle link frame processing capability;

- bundle membership verification;

- loopback detection;

- differential delay measurement;

- coordinated bundle link removal;

- vendor specific extensions; and

- symmetric message exchange between bundle link endpoints.

The bundle and bundle link procedures are described in the following sections.

4.2   Bundle Procedures

4.2.1   General

The bundle procedures operate within the bundle. No messages are exchanged between the bundle peers.

An interface connected by multilink frame relay consists of two bundle endpoints joined by one or more bundle links. The network management function provisions the bundle.   The network management function is beyond the scope of this agreement. The bundle link endpoints must be provisioned consistently with each other (e.g. bundle links originating on one endpoint must terminate on the other – the links cannot terminate on a third bundle endpoint.)

4.2.2   Bundle Management

4.2.2.1   Frame Bearing Capability

A bundle may be provisioned with a minimum acceptable level of operational bandwidth for the bundle. Operational bandwidth is available from a bundle link when BL_ACTIVATE indication is received. The total operational bandwidth available is calculated by adding the operational bandwidth available from each bundle link.

Four classes of bandwidth requirements are shown in Table 2. One class is selected to determine the criteria that trigger activation or deactivation of the emulated physical interface.

When the bundle's operational bandwidth meets the criteria for a selected class, the bundle must send a PH_ACTIVATE to the data link layer.

When the bundle's operational bandwidth fails to meet the criteria for a selected class, the bundle must send a PH_DEACTIVATE to the data link layer.

Class	Description	Criteria for PH_ACTIVATE/PH_DEACTIVATE	
Class A (Default)	Single link	PH_ACTIVATE:	One or more bundle links indicate BL_ACTIVATE.
		PH_DEACTIVATE:	All bundle links indicate BL_DEACTIVATE.
Class B	All links	PH_ACTIVATE:	All bundle links indicate BL_ACTIVATE.
		PH_DEACTIVATE:	Any bundle link indicates BL_DEACTIVATE.
Class C	Threshold	A minimum threshold is provisioned through network management procedures. The threshold represents the minimum number of bundle links that must indicate BL_ACTIVATE.	
		PH_ACTIVATE:	Sufficient bundle links indicate BL_ACTIVATE.
		PH_DEACTIVATE:	Insufficient bundle link indicates BL_DEACTIVATE.
Class D	Custom	PH_ACTIVATE:	Implementation specific.
		PH_DEACTIVATE:	Implementation specific.

Table 2
Class of Bandwidth Requirements

4.2.2.2   Bundle Link Activation

Individual links of the bundle are activated upon receipt of the MB_ADD_LINK request from the bundle's layer management. The bundle issues a BL_ACTIVATE request to the target bundle link. Upon successful completion of bundle link initialization procedures, the BL_ACTIVATE confirmation will be sent to the bundle. Following receipt of the BL_ACTIVATE confirmation from the bundle link, the bundle sends and receives frames on the bundle link.

4.2.2.3   Bundle Link Deactivation

An individual bundle link may send an unsolicited BL_DEACTIVATE indication in the event of a physical interface error, suspected looped back state, or link deactivation from the peer endpoint (See Section 4.3.2.2). Upon receipt of any BL_DEACTIVATE indication, the bundle must not send frames to the unavailable bundle link. When the error condition ends, the bundle link issues a BL_ACTIVATE indication to the bundle. At this time, the bundle may resume sending and receiving frames on the bundle link.

Individual links of the bundle may be deactivated upon receipt of the MB_REMOVE_LINK request from the bundle's layer management. The bundle issues a BL_DEACTIVATE request to the target bundle link. Following this request, the bundle must not send any frames on the bundle link. However, the bundle must continue to accept frames while bundle link deactivation procedures are in progress. Upon completion, the BL_DEACTIVATE confirmation will be sent to the bundle. Upon receipt of this confirmation, the bundle will no longer receive frames from the bundle link.

If the MB_REMOVE_LINK request is received from the bundle's layer management when the bundle link has reported a BL_DEACTIVATE indication, the bundle should send a BL_DEACTIVATE request to the bundle link. This request will prevent the inadvertent re-establishment of bundle link operation when the error condition clears.

4.2.2.4   Use of Timestamp Field

Granularity and interpretation of the Timestamp Information Field is implementation specific.

The primary use of the Timestamp Information Field is to measure the differential delay between bundle links in a bundle.

The Timestamp Information Field may be used to determine if a bundle link has a much more substantial differential delay than other bundle links in the same bundle. The implementing endpoint then can determine if the differential delay is in a tolerable range and decide to remove or keep the bundle link in operation.

4.2.3   Frame Processing

The bundle must support the FRF.12 [3] user-to-network and network-to-network interface fragmentation techniques to format all frames sent to bundle links. The degree of fragmentation supported is implementation specific and must be agreed on by bi-lateral agreement. The default degree of fragmentation is no fragmentation (B=1, E=1). The degree of fragmentation is established via network management procedures. A MFR bundle must not transmit MFR fragmentation frames with (B)egin or (E)nd bits set to zero without a bi-lateral agreement between endpoints The bundle may send the MFR frames on any bundle link that is in the state *up*.

4.2.3.1   Frame Transmission

The data link layer sends frames to the bundle by sending a PH_DATA request. The frame is formatted as described in Section 3.1. Sequence numbers are assigned using the rules provided in Section 5.1 of FRF.12[3].

Frames received from the data link layer may be fragmented when both peer bundles support fragmentation. The fragmentation procedure described in Section 6.1 of FRF.12 is followed. If multiple fragments are created for a single frame received from the data link layer, the fragments may be transmitted on different bundle links.

The bundle may send the MFR frames on any bundle link that is in the state *up*. Implementations should balance the load between the bundle links.

4.2.3.2   Frame Reception

The bundle link sends a BL_DATA indication to the bundle when a MFR frame fragment is received.

The bundle must reassemble the frame fragments using the procedure described in Section 6.2 of FRF.12. The fragments may arrive out of order when transmitted over multiple bundle links. The bundle must preserve the frame order for all frames associated with a single DLCI.

The bundle sends a reassembled frame to the data link layer in a PH_DATA indication.

Re-assembly of a frame requires reception of all fragments. In contrast to FRF.12, the sequence number is not a sufficient indication of fragment loss. This is a consequence of using multiple links. Implementations must provide detection of fragment loss. The mechanism for loss detection is implementation specific. Some examples of loss detection mechanisms are described in Multilink PPP [6].

Implementations utilizing frame assembly timers should carefully select the time interval to avoid premature detection of loss. An example algorithm for creating a time interval (T_i) is provided in Figure 14. In this example, the size used for a "typical" frame is implementation specific.

$$Te = \text{Time required to transmit typical frame over slowest link}$$
$$Ti = 15\,m\sec + Te$$

Figure 14
Example Calculation of Frame Assembly Time Interval

Implementations may utilize longer time intervals to accommodate variations in link propagation characteristics.

4.3   Bundle Link Procedures

4.3.1   General

The Link Integrity Protocol operates on a physical interface (e.g. DS0, DS1, E1(FRF.14[5])). Control messages (e.g., ADD_LINK) never enter the frame relay network.

A message received from the remote endpoint is validated according to the message format described in Section 2.4. Any unrecognized or errored message must be silently discarded unless otherwise specified in the following procedures. Information about invalid messages should be communicated to the layer management function.

The MFR Link Integrity Protocol operates independently on each link of a bundle.

The MFR Link Integrity Protocol operates between the two endpoints of a bundle link. The endpoints function as symmetric peer entities.

MFR implementations must support the MFR Link Integrity Protocol.

4.3.2   Addition of Bundle Link to Bundle Operation

4.3.2.1   Establishment Procedures

These procedures shall be used to establish bundle link operation between two endpoints following receipt of the BL_ACTIVATE request from the bundle.

4.3.2.1.1   Initiating Endpoint

Each bundle link endpoint shall initiate a request for bundle link operation with its peer by transmitting the ADD_LINK message. All existing exception conditions shall be cleared, the retransmission counter shall be reset (counter N_MAX_RETRY is defined in Section 4.3.8.3), and timer T_ACK shall be started (timer T_ACK is defined in Section 4.3.8.2).

A bundle link endpoint shall also transmit the ADD_LINK message and start timer T_ACK whenever the T_HELLO timer expires while in *add sent*, *ack rx*, or *add rx* states.

The T_HELLO timer is used as a pacing timer that is used to send out HELLO messages.

4.3.2.1.2   Responding Endpoint

If a bundle link endpoint receives a valid ADD_LINK message and it is able to enter into the bundle link state *up*, it shall respond with an ADD_LINK_ACK message.

A bundle link endpoint that was removed from bundle operation by a BL_DEACTIVATE.req shall respond to the ADD_LINK message with an ADD_LINK_REJ message. The ADD_LINK_REJ message must contain the LINK_IDLE cause code.

A bundle link endpoint must receive two messages from its peer before transitioning to the bundle link state *up*, as described in the following paragraphs.

One message must be a valid ADD_LINK message sent by the peer. When a valid ADD_LINK message is received, the endpoint enters the bundle link state *add rx*. A transition to the bundle link state *up* is possible following reception of an ADD_LINK_ACK message.

The other message is an ADD_LINK_ACK message sent by the peer in response to an ADD_LINK message sent by the bundle link endpoint. When an ADD_LINK_ACK message is received, the endpoint stops timer T_ACK, starts timer T_HELLO, and enters the bundle link state *ack rx*. A transition to the bundle link state *up* is possible following reception of a valid ADD_LINK message.

The two messages may be received in any order.

Upon reception of both the ADD_LINK message and the ADD_LINK_ACK message, the bundle link endpoint shall:

1. reset timer T_ACK;

2. start timer T_HELLO (timer T_HELLO is defined in Section 4.3.8.1);

3. enter the state *up*; and

4. issue a BL_ACTIVATE.cnf to the bundle.

4.3.2.2   Receiving ADD_LINK_REJ Messages

A bundle link endpoint receiving an ADD_LINK_REJ message shall examine the cause information. If the cause code is UNKNOWN_VENDOR_EXTENSION, the bundle link endpoint shall not include the rejected Vendor Extension Information Field in subsequent ADD_LINK messages. All implementations must be capable of operating without vendor extensions.

Upon reception of all ADD_LINK_REJ messages, the originator of the ADD_LINK message shall:

1. remain in the *add sent or add rx* state;

2. transmit the ADD_LINK message;

3. start timer T_ACK; and

4. send a BL_DEACTIVATE indication to the bundle with the cause code and diagnostics, if diagnostics are present in the message. The bundle should issue a MB_ERROR indication to report the failure to layer management.

Refer to Table 4 for a list of the cause values and the contents of the diagnostic field.

4.3.2.3   Procedure on Expiry of Timer T_ACK

If timer T_ACK expires before a bundle link endpoint receives an ADD_LINK_ACK message, the bundle link endpoint shall transmit the ADD_LINK message by starting timer T_HELLO.

The above procedure continues T_ACK   until one of the following events occurs:

1.   ADD_LINK_ACK message is received;

2.   the bundle issues a BL_DEACTIVATE request primitive; or

3.   the physical layer issues a PH_DEACTIVATE indication.

4.3.2.4   Configuration Mismatch Detection

All bundle links of a MFR must use the same value in the Bundle identification Information Field. The local and remote ends of a bundle link may have different Bundle Identification values as long as the values are consistent with the other bundle links at a specific end of the bundle. Figure 15 illustrates an instance where one device, LONDON, is misconfigured. The link FOX assigned to bundle MARS should be assigned to bundle PLUTO. Device AMSTERDAM will report a bundle consistency error when the ADD_LINK message containing bundle MARS is received on a bundle link assigned to bundle ALPHA when prior ADD_LINK messages contained the bundle identifier PLUTO. Note that if the first ADD_LINK message was received from bundle MARS instead of bundle PLUTO, then the ADD_LINK from one of bundle PLUTO's links would fail. The consistency check is performed based on the first ADD_LINK to arrive. The bundle identification contained in the first ADD_LINK message is always considered the true bundle.

Useful Bundle Identification values include network node identifiers, system serial numbers, and network addresses. A device should use a unique bundle identifier for each bundle if more than one bundle is supported between two devices.
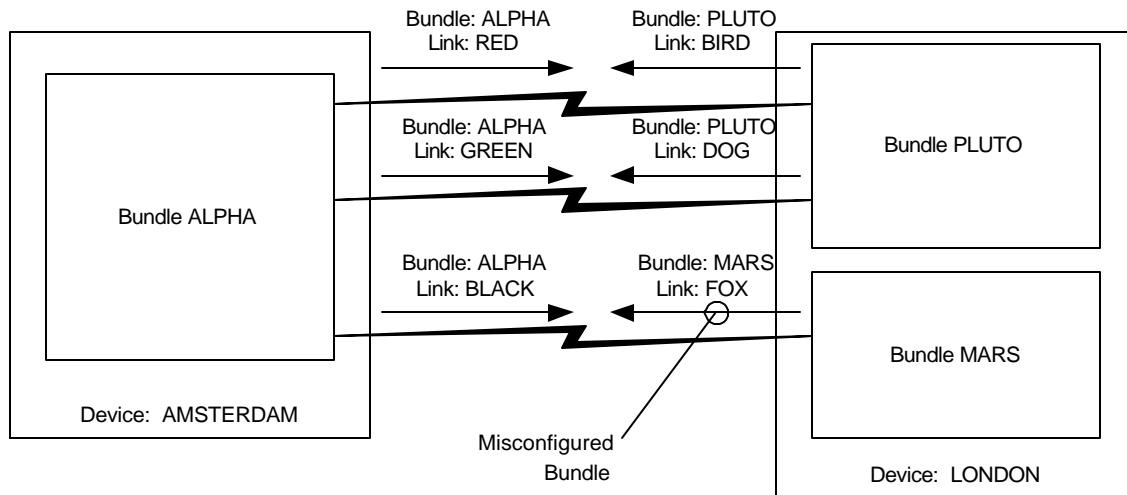
Figure 15
Bundle Configuration Error Example

A received ADD_LINK message containing an inconsistent Bundle Identification value or Endpoint Capabilities encoding shall be processed as an invalid ADD_LINK message using the procedure described in Section 4.3.2.5.

4.3.2.5   Receiving Invalid ADD_LINK Messages

A received ADD_LINK message may be considered invalid when:

1.   the received bundle identification is not consistent with the bundle identification received from the other bundle link of the bundle, or

2.   if the Vendor Extension Information exists and specifies an unknown OUI or sub-code.

A bundle link endpoint receiving an invalid ADD_LINK message shall respond with an ADD_LINK_REJ message.

Information about the invalid ADD_LINK message is communicated to the bundle by issuing a BL_DEACTIVATE indication. Information should include the cause code and diagnostics. The bundle should issue a MB_ERROR indication to report the failure to layer management.

The ADD_LINK_REJ message must also contain the cause for the rejection. Refer to Table 4 for a list of the cause values and the contents of the diagnostic field.

4.3.3   Bundle Link Operation

4.3.3.1   Frame Transfer

The bundle forwards a MFR frame fragment to the bundle link by issuing a BL_DATA request primitive. If the bundle link is in the state *up*, the bundle link issues a PH_DATA request to the physical layer.

MFR frame fragments received on a bundle's physical interface are forwarded to the bundle link via a PH_DATA indication primitive. If the bundle link is in the state *add_rx, up* or *idle pending*, the received frame is forwarded to the bundle for reassembly. The bundle link forwards the fragment by issuing a BL_DATA indication to the bundle.

A MFR frame fragment that is received at a bundle link endpoint in all other states must be silently discarded.

4.3.3.2   Bundle Link Integrity Procedure

These procedures test bundle link integrity during normal operation. A bundle link endpoint shall transmit the HELLO message following expiration of timer T_HELLO. Timer T_ACK shall be started.

A bundle link endpoint receiving a valid HELLO message while in the *up* state shall respond with a HELLO_ACK message. All HELLO messages received when the bundle link is not in the *up* state must be ignored.

A bundle link endpoint receiving a valid HELLO_ACK message shall:

1. reset RETRY counter;
2. stop timer T_ACK; and
3. start timer T_HELLO.

4.3.3.3   Procedure on Expiry of Timer T_ACK

Upon expiration of timer T_ACK the bundle link endpoint shall test the RETRY count.

If the RETRY count is less than N_MAX_RETRY, the bundle link endpoint shall:

1. send HELLO message;
2. increment RETRY count by one; and
3. start timer T_ACK;

If RETRY count is equal to N_MAX_RETRY, the bundle link endpoint shall:

1. send ADD_LINK message;
2. reset RETRY count;
3. start timer T_ACK; and
4. enter the state *add sent*.

4.3.4   Removal of Bundle Link from Bundle Operation

4.3.4.1   Release Procedures

These procedures remove a bundle link from bundle operation following receipt of the BL_DEACTIVATE request from the bundle.

4.3.4.1.1   Initiating Endpoint

When the BL_DEACTIVATE request is received from the bundle while the bundle link endpoint is in the *add sent*, *ack rx*, and *add rx* states, the bundle link endpoint shall:

1. transmit the REMOVE_LINK message;

2. clear all existing exception conditions;

3. reset RETRY counter;

4. start timer T_ACK;

5. send a BL_DEACTIVATE confirmation to the bundle with a cause of LINK_IDLE; and

6. enter the state *idle*.

When the BL_DEACTIVATE request is received in the *up* state, the bundle link endpoint shall:

1. transmit the REMOVE_LINK message;

2. clear all existing exception conditions;

3. reset RETRY counter;

4. start timer T_ACK;

5. enter the state *idle pending*.

> NOTE: While in the state *idle pending* all MFR fragmentation frames received on the bundle link must be forwarded to the bundle via the BL_DATA indication. The bundle must not transmit any MFR fragmentation frames on a bundle link that is in the state *idle pending*.

4.3.4.1.2   Responding Endpoint

A bundle link receiving a REMOVE_LINK message, in states: *add sent*, *ack rx*, *add rx*, or *up*, shall:

1. respond with a REMOVE_LINK_ACK message;

2. reset RETRY count;

3. stop timer T_ACK;

4. start timer T_HELLO to resume ADD_LINK;

5. send a BL_DEACTIVATE indication to the bundle, with a cause code of LINK_IDLE; and

6. enter the state *add sent*.

A bundle link endpoint receiving a REMOVE_LINK message when in the state *idle pending* shall:

1. respond with a REMOVE_LINK_ACK message;

2. send a BL_DEACTIVATE indication to the bundle, with a cause code of LINK_IDLE; and

3. enter the state *idle*.

A bundle link endpoint receiving a REMOVE_LINK message when in the state *idle* shall respond with a REMOVE_LINK_ACK message and remain in the state *idle*.

Upon reception of a REMOVE_LINK_ACK message, the initiator of the REMOVE_LINK message shall:

1. stop timer T_ACK;

2. send a BL_DEACTIVATE indication to the bundle, with a cause code of LINK_IDLE; and

3. enter the state *idle*.

4.3.4.2   Procedure on Expiry of Timer T_ACK

Upon expiration of timer T_ACK while the bundle link endpoint is in the state *idle pending,* the bundle link endpoint shall test RETRY count.

If the RETRY count is less than N_MAX_RETRY, the bundle link endpoint shall:

1. send REMOVE_LINK message;

2. increment RETRY count; and

3. start timer T_ACK;

If the RETRY count is equal to N_MAX_RETRY, the bundle link endpoint shall:

1. reset RETRY count;

2. send a BL_DEACTIVATE indication to the bundle, with a cause code of LINK_IDLE; and

3. enter the state *idle*.

4.3.5   Loss of Physical Layer When Administratively Up

Upon receipt of a PH_DEACTIVATE indication from the physical layer, a bundle link endpoint in the *add sent, ack rx, add rx,* or *up* states shall:

1. stop all timers;

2. send a BL_DEACTIVATE indication to the bundle, with a cause code of LINK_DOWN; and

3. enter the state *down.*

Upon restoration of the physical layer, the bundle link endpoint shall:

1. send ADD_LINK message;

2. start timer T_ACK (T_HELLO will commence on expiration of T_ACK); and

3. enter the state *add sent*.

4.3.6   Loss of Physical Layer When Administratively Down

Upon receipt of a PH_DEACTIVATE indication from the physical layer , a bundle link endpoint in the *idle pending* or *idle* states shall:

1. stop all timers;

2. reset RETRY count;

3. send a BL_DEACTIVATE indication to the bundle, with a cause code of LINK_DOWN; and

4. enter the state *down idle.*

Upon restoration of the physical layer, the bundle link endpoint shall enter the *idle* state.

4.3.7   Looped-back Link Detection Procedure

Looped-back bundle link is detected through examination of the Magic Number Information Field contained in every message. An endpoint must select the magic number based on the criteria described in Section 6.4 of PPP[10].

Any message received, which contains a magic number identical to the magic number last transmitted by the bundle link endpoint, may indicate a loopback condition. This message must be silently discarded. The local endpoint must select a different magic number for use in the transmission of subsequent messages.

Information about the possible looped-back bundle link should be communicated to the bundle when the bundle link becomes deactivated due to the suspected looped-back condition. The bundle should report this to layer management via the MB_ERROR indication primitive.

4.3.8   System Parameters

The system parameters that apply to the MFR Link Integrity Protocol are listed below. Table 3 lists the default values and acceptable ranges for these system parameters.

4.3.8.1   Timer T_HELLO

The T_HELLO timer controls the rate at which HELLO messages are sent. Following a period of T_HELLO duration, a HELLO message is transmitted according to the procedures described in Section 4.3.2.1.

4.3.8.2   Timer T_ACK

The T_ACK timer specifies the maximum time period to wait for an ADD_LINK_ACK, HELLO_ACK or REMOVE_LINK_ACK message.

4.3.8.3   Maximum Retransmission Count N_MAX_RETRY

The N_MAX_RETRY count limits the number of retransmission attempts for consecutive HELLO or REMOVE_LINK messages following expiration of timer T_ACK.

Parameter	Default Value	Minimum Value	Maximum Value
Timer T_HELLO	10 seconds	1 second	180 seconds
Timer T_ACK	4 seconds	1 second	10 seconds
Count N_MAX_RETRY	2	1	5

Table 3
System Parameters

4.3.9   Error Conditions

Each error condition is identified by a cause value that is included in the Cause Information Field. Some cause values result in the inclusion of additional diagnostic information in the variable length diagnostic field of the Cause Information Field. If the diagnostic field is not used it is empty. The cause values are defined in Table 4.

Error	Description	Cause	Diagnostic Field Usage
INCONSISTENT_BUNDLE	Possible configuration mismatch detected.	1	Expected Bundle Identification value as provisioned at the endpoint sending the Cause Information Field. Refer to Note 2, Figure 8 for the format of the field.
UNKNOWN_VENDOR	Unrecognized OUI in a received Vendor Specific Information Field.	2	None
LINK_IDLE	The bundle link is not operational.	3	None
LINK_DOWN	The bundle link physical layer is down.	4	None
DIFFERENTIAL_DELAY	The bundle link differential delay exceeds the maximum allowed.	5	None
LOOPBACK_DETECTED	The bundle link has detected a potential loopback condition.	6	None
OTHER	Generic failure cause described by text in diagnostic field.	7	Textual description of failure cause. Refer to Note 2, Figure 8 for the format of the field.

Table 4
Cause Values

A   Informative Annex – Bundle Link Protocol State Machine

The purpose of this annex is to provide one example of a state machine representation of the bundle link procedures, to assist in the understanding of this agreement. This example of the bundle link procedures may not meet all the requirements specified in the procedure section (Section 4). The text description of the procedures is definitive.

A simplified version (for clarity) of the state machine for the MFR Link Integrity Protocol is shown in Figure 16. The initial state is *down*. The picture is divided into four quadrants for clarity:

|	
Physically Up, Administratively Up	The normal operational mode for a bundle link.
Physically Up, Administratively Down	Bundle link is not participating in bundle operation.
Physically Down, Administratively Up	The bundle link does not support frame transmission.
Physically Down, Administratively Down	Bundle link is not participating in bundle operation and the link is physically unable to support frame transmission.
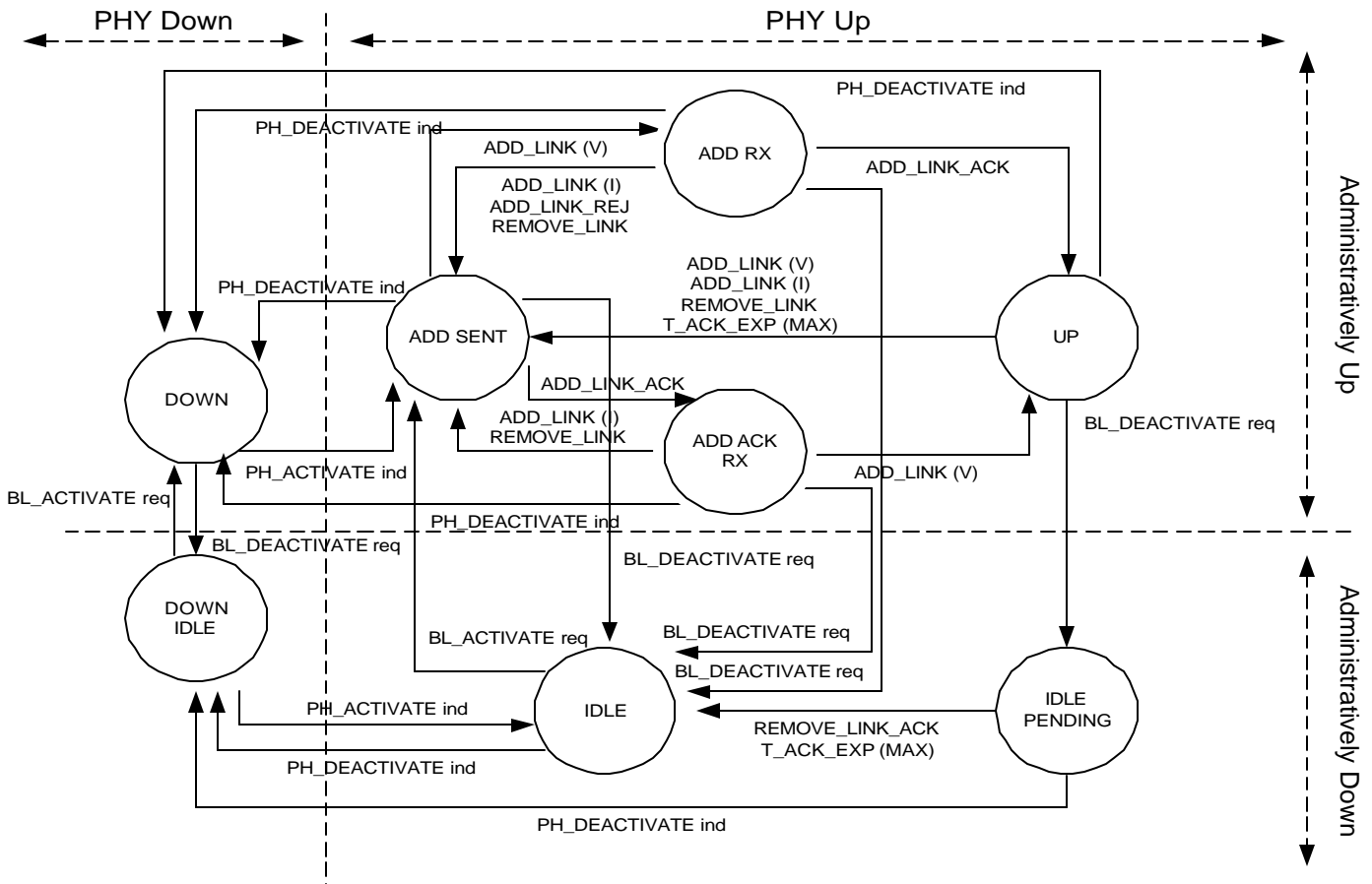
Figure 16
MFR Link Integrity Protocol State Machine

Table 5, Table 6, and Table 7 describe the state transitions. Refer to the following sections for a description of the states, events, and actions. A transition to the next state is indicated by the "=>" symbol followed by the next state name. A "-" symbol indicates an unexpected event and state combination.

Event	*add sent*	*ack rx*	*add rx*	*up*	*idle pending*	*idle*
ADD_LINK (Valid)	Send ADD_LINK_ACK => *add rx*	stop T_ACK send ADD_LINK_ACK start T_HELLO BL_ACTIVATE.cnf => up	send ADD_LINK_ACK => *add rx*	send ADD_LINK start T_ACK => *add sent*	send ADD_LINK_REJ => *idle pending*	send ADD_LINK_REJ => *idle*
ADD_LINK (Invalid)	BL_DEACTIVATE.ind send ADD_LINK_REJ => *add sent*	BL_DEACTIVATE.cnf send ADD_LINK_REJ send ADD_LINK start T_ACK => *add sent*	BL_DEACTIVATE.cnf send ADD_LINK_REJ send ADD_LINK start T_ACK => *add sent*	BL_DEACTIVATE.cnf send ADD_LINK_REJ send ADD_LINK start T_ACK => *add sent*	-	-
ADD_LINK_ACK	Stop T_ACK Start T_HELLO => *ack rx*	start T_HELLO => *ack rx*	stop T_ACK start T_HELLO BL_ACTIVATE.cnf => *up*	-	=> *idle pending*	=> *idle*
ADD_LINK_REJ	BL_DEACTIVATE.cnf send ADD_LINK Start T_ACK => *add sent*	BL_DEACTIVATE.cnf send ADD_LINK start T_ACK => *add sent*	BL_DEACTIVATE.cnf send ADD_LINK start T_ACK => *add rx*	-	=> *idle pending*	=> *idle*
HELLO	=> *add sent*	=> *ack rx*	=> *add rx*	send HELLO_ACK => *up*	=> *idle pending*	=> *idle*
HELLO_ACK	=> *add sent*	=> *ack rx*	=> *add rx*	stop T_ACK start T_HELLO RETRY = 0 => *up*	=> *idle pending*	=> *idle*
REMOVE_LINK	Stop T_ACK send REMOVE_LINK_ACK start T_HELLO RETRY = 0 => *add sent*	stop T_ACK send REMOVE_LINK_ACK start T_HELLO RETRY = 0 => *add sent*	stop T_ACK send REMOVE_LINK_ACK start T_HELLO RETRY = 0 => *add sent*	stop T_ACK send REMOVE_LINK_ACK start T_HELLO RETRY = 0 => *add sent*	send REMOVE_LINK_ACK BL_DEACTIVATE.cnf => *idle*	send REMOVE_LINK_ACK => *idle*
REMOVE_LINK_ACK	=> *add sent*	-	-	-	stop T_ACK BL_DEACTIVATE.ind => *idle*	=> *idle*
T_HELLO_EXP	Send ADD_LINK start T_ACK => *add sent*	send ADD_LINK start T_ACK => *ack rx*	send ADD_LINK start T_ACK => *add rx*	send HELLO start T_ACK => *up*	-	-
T_ACK_EXP RETRY < N_MAX_RETRY	Start T_HELLO => *add sent*	start T_HELLO => *ack rx*	start T_HELLO => *add rx*	send HELLO RETRY = RETRY + 1 start T_ACK => *up*	send REMOVE_LINK RETRY = RETRY + 1 start T_ACK => *idle pending*	-
T_ACK_EXP RETRY = N_MAX_RETRY	-	-	-	send ADD_LINK start T_ACK RETRY = 0 => *add sent*	RETRY = 0 BL_DEACTIVATE.ind => *idle*	-

Table 5
MFR Link Integrity Protocol State Transitions – Normal States – Part 1

A "-" symbol indicates an unexpected event and state combination.

Event	*add sent*	*ack rx*	*add rx*	*Up*	*idle pending*	*idle*
PH_DEACTIVATE.ind	stop T_ACK stop T_HELLO BL_DEACTIVATE.cnf => *down*	stop T_ACK stop T_HELLO BL_DEACTIVATE.cnf => *down*	stop T_ACK stop T_HELLO BL_DEACTIVATE.cnf => *down*	stop T_ACK stop T_HELLO RETRY = 0 BL_DEACTIVATE.cnf => *down*	stop T_ACK, RETRY = 0 BL_DEACTIVATE.cnf => *down_idle*	=> *down_idle*
PH_ACTIVATE.ind	-	-	-	-	-	-
PH_DATA.ind	-	-	-	BL_DATA.ind => *up*	BL_DATA.ind => *idle pending*	-
BL_ACTIVATE.req	-	-	-	-	-	send ADD_LINK start T_ACK => *add sent*
BL_DEACTIVATE.req	stop T_HELLO stop T_ACK send REMOVE_LINK BL_DEACTIVATE.cnf => *idle*	stop T_HELLO stop T_ACK send REMOVE_LINK BL_DEACTIVATE.cnf => *idle*	stop T_HELLO stop T_ACK send REMOVE_LINK BL_DEACTIVATE.cnf => *idle*	send REMOVE_LINK stop T_HELLO start T_ACK RETRY = 0 => *idle pending*	-	-
BL_DATA.req	-	-	-	send DATA => *up*	-	-

Table 6
MFR Link Integrity Protocol State Transitions – Normal States – Part 2
A "-" symbol indicates an unexpected event and state combination.

Event	*Down*	*down idle*
ADD_LINK (Valid)	-	-
ADD_LINK (Invalid)	-	-
ADD_LINK_ACK	-	-
ADD_LINK_REJ	-	-
HELLO	-	-
HELLO_ACK	-	-
REMOVE_LINK	-	-
REMOVE_LINK_ACK	-	-
T_HELLO_EXP	-	-
T_ACK_EXP RETRY < N_RETRY_MAX	-	-
T_ACK_EXP RETRY = N_RETRY_MAX	-	-
PH_DEACTIVATE.ind	=> *down*	-
PHY_ACTIVATE.ind	send ADD_LINK start T_ACK => *add sent*	=> *idle*
PH_DATA	-	-
BL_ACTIVATE.req	=> down	=> *down*
BL_DEACTIVATE.req	BL_DEACTIVATE.cnf => *down idle*	BL_DEACTIVATE.cnf => *down idle*
BL_DATA.req	-	-

Table 7
MFR Link Integrity Protocol State Transitions – Down States
A "-" symbol indicates an unexpected event and state combination.

A.1   States

The MFR Link Integrity Protocol state machine states are defined as follows:

down   The bundle link is physically incapable of frame operation. The bundle link is not active in bundle operation.

down idle   The bundle link is physically incapable of frame operation while also administratively removed from bundle operation.

add sent   The bundle link is available for frame operation and contact with the peer initiated.

ack rx   The bundle link has received the ADD_LINK_ACK from the peer and now awaits receipt of an ADD_LINK message.

add rx   The bundle link has received and accepted an ADD_LINK message from the peer and now awaits receipt of an ADD_LINK_ACK message. Frames may be received from the peer when the bundle link is in this state.

up   The bundle link is fully operational and joined to bundle operation.

idle pending   The bundle link is in the process of being removed from bundle operation. Frames received from the peer are processed until the peer acknowledges the REMOVE_LINK message. No additional frames are sent to the peer.

idle   The bundle link is removed from bundle operation.

A.2   Events

The MFR Link Integrity Protocol state machine events are generated by messages received from the peer, timer expirations, physical layer indications, and layer management primitives. The events are described as follows:

ADD_LINK (valid)   A valid ADD_LINK message is received and validated. Validation includes the bundle identification and magic number.

ADD_LINK (invalid)   An ADD_LINK message is received with invalid bundle identification or a magic number that indicates potential loopback.

ADD_LINK_ACK   An ADD_LINK_ACK message is received from the peer.

ADD_LINK_REJ   An ADD_LINK_REJ message is received from the peer.

HELLO   A HELLO message is received from the peer.

HELLO_ACK   A HELLO_ACK is received is received from the peer.

REMOVE_LINK   A REMOVE_LINK message is received from the peer.

REMOVE_LINK_ACK   A REMOVE_LINK_ACK message is received from the peer.

T_HELLO_EXP   The HELLO polling timer T_HELLO has expired.

T_ACK_EXP
RETRY < N_MAX_RETRY   The acknowledgment response timer T_ACK has expired with additional retry attempts available.

T_ACK_EXP
RETRY = N_MAX_RETRY   The acknowledgment response timer T_ACK has expired and all retry attempts are now exhausted.

PH_DEACTIVATE ind	The physical layer does not support transmission of frames.
PH_ACTIVATE ind	The physical layer now supports transmission of frames.
PH_DATA ind	A DATA frame is received from the peer.
BL_ACTIVATE req	Bundle requests addition of the bundle link to bundle operation.
BL_DEACTIVATE req	Bundle requests removal of the bundle link from bundle operation.
BL_DATA req	Bundle requests transmission of a frame fragment.

A.3   Actions

The MFR Link Integrity Protocol state machine actions are defined as follows:

start T_HELLO	Start the T_HELLO polling interval timer.
stop T_HELLO	Stop the T_HELLO polling interval timer.
start T_ACK	Start the T_ACK acknowledgment response timer.
stop T_ACK	Stop the T_ACK acknowledgment response timer.
send ADD_LINK	Transmit the ADD_LINK message to the peer.
send ADD_LINK_ACK	Transmit the ADD_LINK_ACK message to the peer.
send ADD_LINK_REJ	Transmit the ADD_LINK_REJ message to the peer.
send HELLO	Transmit the HELLO message to the peer.
send HELLO_ACK	Transmit the HELLO_ACK message to the peer.
send REMOVE_LINK	Transmit the REMOVE_LINK message to the peer. Sent to reject an invalid HELLO message or to remove a bundle link from a bundle.
send REMOVE_LINK_ACK	Transmit the REMOVE_LINK_ACK message to the peer. Sent following last frame in transmit.
send DATA	Transmit the frame fragment to the peer.
RETRY = 0	Reset the retransmission counter to a value of zero.
RETRY = RETRY + 1	Increment the retransmission counter by one.
BL_DATA ind	Issue a bundle layer primitive indication of frame receipt.
BL_ACTIVATE cnf	Issue an activation confirmation to the bundle.
BL_DEACTIVATE cnf	Issue a deactivation confirmation to the bundle
BL_DEACTIVATE ind	Issue a deactivation indication to the bundle. Deactivation may be in response to a request from the bundle or as the result of a failure. The primitive provides the cause information to discriminate between deactivation triggers.

FRF.16 Errata Sheet

Frame Relay Forum Technical Committee
October 1999

ERRATA

Figure 7 of Section 3.2 is replaced with the following corrected figure. The purpose of the replacement is to align the Information Field Type encodings with the bit encodings shown in the figures contained within Sections 3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5, and 3.4.6.

			Bits					
8	7	6	5	4	3	2	1	Octets
Type								1 (Note 1)
Length								2 (Note 2)
Data								3 - N

NOTES:
1. The following information field type encodings are used:
   1 Bundle identification
   2 Link identification
   3 Magic number
   4 *Reserved*
   5 Timestamp information
   6 Vendor extension
   7 Cause

2. Length includes the Type, Length, and Data sub-fields.

Figure 17
MFR Link Integrity Protocol Information Field Format

FRF.16 Errata Sheet

Frame Relay Forum Technical Committee
January 2000

ERRATA

Section 3.4

Figure 7 of Section 3.4 is replaced with the following corrected figure. The purpose of the replacement is to align the Information Field Type encodings with the bit encodings shown in the figures contained within Sections 3.4.1, 3.4.2, 3.4.3, 3.4.4, 3.4.5, and 3.4.6.

			Bits					
8	7	6	5	4	3	2	1	Octets
Type								1 (Note 1)
Length								2 (Note 2)
Data								3 - N

NOTES:
1. The following information field type encodings are used:
   1 Bundle identification
   2 Link identification
   3 Magic number
   4 *Reserved*
   5 Timestamp information
   6 Vendor extension
   7 Cause

2. Length includes the Type, Length, and Data sub-fields.

Figure 18
MFR Link Integrity Protocol Information Field Format

Section 4.3.2.5

In Section 4.3.2.5, a new reason is added for an ADD_LINK message to be invalid:

3.   if the bundle link state is *up*.

Section 4.3.9

In Section 4.3.9, a new error reason is added to Table 4:

Error	Description	Cause	Diagnostic Field Usage
UNEXPECTED_ADDLINK	An ADD_LINK message was received when the Bundle Link was in *up* state.	8	None

Table 4
Cause Values

Annex A

When in the *up* state there is no longer any ADD_LINK message that is valid when received. The contents of this box in Table 5 should be replaced with a "-" indicating an invalid combination.

Event	*up*
ADD_LINK (Valid)	-

Table 5
MFR Link Integrity Protocol State Transitions – Normal States – Part 1

FRF.16 Errata Sheet

Frame Relay Forum Technical Committee
April 2000

ERRATA

4.4   Section 4.3.2.2

The following changes should be made to section 4.3.2.2. The changes are shown in red and strikeout.

4.3.2.2   Receiving ADD_LINK_REJ Messages

A bundle link endpoint receiving an ADD_LINK_REJ message shall examine the cause information. If the cause code is UNKNOWN_VENDOR_EXTENSION, the bundle link endpoint shall not include the rejected Vendor Extension Information Field in subsequent ADD_LINK messages. All implementations must be capable of operating without vendor extensions.

Upon reception of all ADD_LINK_REJ messages, the originator of the ADD_LINK message shall:

5. remain in the *add sent or add rx* state;

6. transmit the corrected ADD_LINK message. If the ADD_LINK message cannot be corrected then the sending of the ADD_LINK should be delayed by the T_HELLO timer. Implementations of MFR that don't support vendor extensions can safely assume that the ADD_LINK_REJ message is not correctable.

7. Stop timer T_ACK; start timer ~~T_ACK~~ T_HELLO; and

8. send a BL_DEACTIVATE indication to the bundle with the cause code and diagnostics, if diagnostics are present in the message. The bundle should issue a MB_ERROR indication to report the failure to layer management.

Refer to Table 4 for a list of the cause values and the contents of the diagnostic field.

Note: An example of where the ADD_LINK message probably cannot be corrected is when the cause code is INCONSISTENT_BUNDLE. An example of where the ADD_LINK message probably can be corrected is when the cause is UNKNOWN_VENDOR_EXTENSION. In the latter case the corrected ADD_LINK message will not have the Vendor Extension Information Field.

4.4.1.1   Changes to Table 5 in Annex A

The changes to Table 5 in Annex A are shown in red and strikeout.

Event	*Add sent*	*ack rx*	*add rx*	*up*	*idle pending*	*idle*
ADD_LINK_REJ	BL_DEACTIVATE.cnf ~~send ADD_LINK~~ ~~Start T_ACK~~ Stop T_ACK Start T_HELLO *=> add sent*	BL_DEACTIVATE.cnf ~~send ADD_LINK~~ ~~Start T_ACK~~ Stop T_ACK Start T_HELLO *=> add sent*	BL_DEACTIVATE.cnf ~~send ADD_LINK~~ ~~Start T_ACK~~ Stop T_ACK Start T_HELLO *=> add rx*	-	*=> idle pending*	*=> idle*