802.1ag™

IEEE Standard for Local and metropolitan area networks—

Virtual Bridged Local Area Networks

Amendment 5: Connectivity Fault Management

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

17 December 2007

IEEE Std 802.1ag™-2007
(Amendment to
IEEE Std 802.1Q™-2005
as amended by
IEEE Std 802.1ad™-2005
and IEEE Std 802.1ak™-2007)

IEEE Std 802.1ag™-2007
(Amendment to
IEEE Std 802.1Q™-2005
as amended by
IEEE Std 802.1ad™-2005
and IEEE Std 802.1ak™-2007)

IEEE Standard for Local and metropolitan area networks—

Virtual Bridged Local Area Networks

Amendment 5: Connectivity Fault Management

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 27 September 2007

IEEE SA-Standards Board

Abstract: This amendment specifies protocols, procedures, and managed objects to support connectivity fault management. These allow discovery and verification of the path, through bridges and LANs, taken for frames addressed to and from specified network users. Connectivity faults can be detected and isolated to an individual bridge or LAN.
Keywords: error detection, fault management, LANs, local area networks, MAC Bridges, MANs, metropolitan area networks, OAM, transparent bridging, VLANs

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-0-7381-5689-7 STD95724
Print: ISBN 978-0-7381-5690-3 STDPD95724

Introduction

This introduction is not part of IEEE Std 802.1ag-2007, IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks—Amendment 5: Connectivity Fault Management.

The MAC Bridge standardization activities that resulted in the development of IEEE Std 802.1D™ introduced the concept of Filtering Services in Bridged Local Area Networks, mechanisms whereby filtering information in such LANs can be acquired and held in a Filtering Database, as well as support for expedited data, dynamic filtering of Group MAC addresses through the GMRP protocol, and the GARP protocol that supports GMRP and can support other applications.

IEEE Std 802.1Q-2005, a revision of IEEE Std 802.1Q-1998, defines mechanisms that allow the implementation of Virtual Bridged Local Area Networks, including:

a) Virtual LAN Services;
b) The operation of the Forwarding Process that is required;
c) The structure of the Filtering Database that is required;
d) The nature of the protocols and procedures that are required in order to provide Virtual LAN services, including the definition of the frame formats used to represent VLAN identification information, and the procedures used in order to insert and remove VLAN identifiers and the headers in which they are carried;
e) The ability to support end-to-end signaling of priority information regardless of the intrinsic ability of the underlying MAC protocols to signal priority information;
f) The GARP VLAN Registration Protocol (GVRP) that allows distribution and registration of VLAN membership information (the protocol described makes use of the GARP protocol defined in IEEE Std 802.1D);
g) The management services and operations that are required in order to configure and administer networks;
h) The ability to restrict Dynamic Group and VLAN registration based on the contents of static filtering entries;
i) VLAN classification according to link layer protocol type;
j) Support for VLANs carried over multiple Spanning Tree instances.

IEEE Std 802.1ad-2005, an amendment to IEEE Std 802.1Q-2005, defines mechanisms that allow the implementation of Provider Bridged Networks, including:

k) The differentiation between Service VLANs (S-VLANs), identified by a new S-VLAN tag, and customer VLANs (C-VLANs), identified by the IEEE 802.1Q tag;
l) The encapsulation of C-VLAN tags within S-VLAN tags, in order to allow a service provider to carry some or all of a customer's C-VLANs within a single S-VLAN;
m) The provision of two levels of drop precedence separate from the priority levels of IEEE Std 802.1D, and the replacement of the Canonical Format Indicator of the C-VLAN tag by a Drop Eligible Indicator in the S-VLAN tag to carry the drop precedence;
n) The passing of certain of the customers' control PDUs, including IEEE 802.1AB LLDP PDUs and IEEE 802.1Q Bridge PDUs, transparently through the Provider Bridged Network as ordinary data;
o) The reassignment of MAC addresses used for control PDUs, including BPDUs, for use by Provider Bridges;
p) The definition of C-VLAN components, added to Provider Bridge Ports to provide the functions necessary to support the allocation of C-VLANs to S-VLANs, to map priorities between networks, and to interact with the customers' control protocols; and
q) The managed objects necessary to control these functions.

This standard provides Connectivity Fault Management (CFM) capabilities useful to Virtual Bridged Local Area Networks for detecting, isolating, and reporting connectivity faults. It is aimed primarily at Provider

Bridged Networks, but is useful also for C-VLAN networks. The mechanisms defined in this standard include:

r) The ability to configure multiple nested Maintenance Domains over a Bridged Network or a network of Bridged Networks, each potentially managed by a different administrative organization;
s) The ability to configure Maintenance Associations (MAs), each identified with a single Maintenance Domain, and on any given Bridge, a set of VLANs;
t) The protocols, procedures, and CFM Protocol Data Unit (CFM PDU) formats required to detect, isolate, and report connectivity faults;
u) The ability, within an MA, to configure and manage Maintenance Points (MPs) that generate and respond to CFM PDUs; and
v) The ability to command MPs to perform certain fault isolation operations, and to inspect the results.

CFM is not to be confused with the Ethernet OAM capabilities described in IEEE Std 802.3™, Clause 57. IEEE 802.3 OAM is concerned with detecting and reporting faults on a single point-to-point IEEE 802.3 LAN. IEEE Std 802.1ag-2007 is concerned with detecting, isolating, and reporting connectivity faults spanning networks comprising multiple LANs, including LANs other than IEEE 802.3 media.

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material. Information on the current revision state of this and other IEEE 802 standards can be obtained from

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Notice to users

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. A patent holder or patent applicant has filed a statement of assurance that it will grant licenses under these rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses. Other Essential Patent Claims may exist for which a statement of assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions are reasonable or non-discriminatory. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the following were participants in the IEEE 802.1 Working Group:

Anthony A. Jeffree, *Chair*

Paul Congdon, *Vice Chair*

Stephen R. Haddock, *Interworking Task Group Chair*

Norman W. Finn, *Editor-in-Chief*

David V. Elie-Dit-Cosaque, Dinesh Mohan,
Oscar Rodriguez, Ali Sajassi, *Assistant Editors*

Alexei Beliaev
Jean-Michel Bonnamy
Mike Borza
Paul Bottorff
Rudolf Brandner
Jim Burns
Dirceu Cavendish
Frank Chao
Uri Cummings
Russell Dietz
Linda Dunbar
Hesham Elbakoury
Donald W. Fedyk
Felix Feifei Feng
David Frattura
John Fuller
Geoffrey Garner
Anoop Ghanwani
Ken Grewal
Romain Insler
Ran Ish-Shalom
Vipin Jain
Hal Keen
Yongbum Kim
Mike Ko
Bruce Kwan
Kari Laihonen
Yannick Le Goff
David Martin
John Messenger
Hiroshi Ohta
Don Pannell
Glenn Parsons
Ken Patton
Haim Porat
Ray Qiu
Karen Randall
Robert Roden
Josef Roese
Allyn Romanow
Dan Romascanu
Jessy V. Rouyer
Eric Ryu
Panagiotis Saltsidis
Sam Sambasivan
John M. Sauer
Michael J. Seaman
Koichiro Seto
Curtis Simonson
Nurit Sprecher
Kevin B. Stanton
Robert Sultan
Muneyoshi Suzuki
Francois Tallet
Michael Johas Teener
John Terry
Patricia A. Thaler
Dennis Volpano
Manoj Wadekar
Bert Wijnen
Ludwig Winkel
Michael D. Wright

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Toru Aihara
Thomas Alexander
Butch Anton
Hugh Barrass
Tomo Bogataj
Matthew Burnburg
Edward J. Carley
James T. Carlo
Juan C. Carreon
Keith Chow
Bryan P. Cook
Tommy P. Cooper
Wael W. Diab
Russell S. Dietz
Thomas J. Dineen
Sourav K. Dutta
David V. Elie-Dit-Cosaque
Donald W. Fedyk
Norman W. Finn
C. J. Fitzgerald
Yukihiro Fujimoto
Devon L. Gayle
Michael D. Geipel
Randall C. Groves
C. G. Guy
Stephen R. Haddock
John F. Hawkins
David Hunter
Atsushi Ito
Raj Jain
Anthony A. Jeffree
Michael D. Johas Teener
Peter G. Jones
Bobby Jose
Shinkyo Kaku
Junghong Kao
Piotr Karocki
Stuart J. Kerry
Lior Khermosh
Yongbum Kim
Cees Klik
Thomas M. Kurihara

Figures

Tables

Contents

IEEE Standard for
Local and metropolitan area networks —

Virtual Bridged Local Area Networks

Amendment 5: Connectivity Fault Management

NOTE—The editing instructions contained in this amendment define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italic.*** Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) or underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.

1. Overview

Insert the following paragraph after the initial paragraphs in Clause 1:

This standard specifies protocols and protocol entities within the architecture of VLAN-aware Bridges that provide capabilities for detecting, verifying, and isolating connectivity failures in Virtual Bridged Local Area Networks. These capabilities can be used in networks operated by multiple independent organizations, each with restricted management access to each other's equipment.

1.1 Scope

Insert the following at end of 1.1, renumbering the list items so that they follow in order from those in the existing text.

This standard specifies protocols, procedures, and managed objects to support connectivity fault management. These allow discovery and verification of the path, through bridges and LANs, taken for frames addressed to and from specified network users, and support detection and isolation of a connectivity fault to a specific bridge or LAN. To this end it:

a) Defines Maintenance Domains, Maintenance Associations, their constituent Maintenance Points, and the managed objects required to create and administer them;
b) Describes the protocols and procedures used by Maintenance Points to detect and diagnose connectivity faults within a Maintenance Domain.

2. Normative references

Change the introductory paragraph as shown, add the references in alphanumeric order, and update the footnote numbering:

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in the text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

IEEE Std 802a™-2003, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture—Amendment 1: Ethertypes for Prototype and Vendor-Specific Protocol Development.

IETF RFC 1035 (STD 13), Domain Names: Implementation and Specification.[1]

IETF RFC 2119 (BCP 14), Key Words for Use in RFCs to Indicate Requirement Levels.

IETF RFC 2578 (STD 58), Structure of Management Information Version 2 (SMIv2).

IETF RFC 2579 (STD 58), Textual Conventions for SMIv2.

IETF RFC 2580 (STD 58), Conformance Statements for SMIv2.

IETF RFC 2685 (Proposed standard), Virtual Private Networks Identifier.

IETF RFC 2863 (Draft standard), The Interfaces Group MIB.

IETF RFC 3413 (STD 62), Simple Network Management Protocol (SNMP) Applications.

IETF RFC 3414 (STD 62), User-based Security Model (USM) for Version 3 of the Simple Network Management Protocol (SNMPv3).

IETF RFC 3415 (STD 62), View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP).

IETF RFC 3417 (STD 62), Transport Mappings for the Simple Network Management Protocol (SNMP).

IETF RFC 3418 (STD 62), Management Information Base (MIB) for the Simple Network Management Protocol (SNMP).

IETF RFC 3419 (Proposed standard), Textual Conventions for Transport Addresses.

IETF RFC 4188 (Proposed standard), Definitions of Managed Objects for Bridges.

IETF RFC 4318 (Proposed standard), Definitions of Managed Objects for Bridges with Rapid Spanning Tree Protocol.

IETF RFC 4363 (Proposed standard), Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering, and Virtual LAN Extensions.

IETF RFC 4789 (Proposed standard), Simple Network Management Protocol (SNMP) over IEEE 802 Networks.

[1]IETF documents are available at http://www.ietf.org/.

ITU-T G.806 (2006), Characteristics of Transport Equipment—Description Methodology and Generic Functionality.[2]

ITU-T X.690 (2002), Information Technology—ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER).

ITU-T Y.1731 (2006), OAM Functions and Mechanisms for Ethernet-based Networks.

[2]ITU-T publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (http://www.itu.int/).

3. Definitions

Change the following definition, but do not change the NOTE following this definition.

~~**3.65 service instance:** An instance of the MAC Service, providing the service at a number of service access points. The term "service instance" is often used to refer to the service and connectivity provided by a service provider to a customer.~~

3.65 service instance: A service instance is a set of Service Access Points (SAPs) such that a Data.Request primitive presented to one SAP can result in a Data.Indication primitive occurring at one or more of the other SAPs in that set. In the context of operators and customers, a particular customer is given access to all of the SAPs of such a set by the operator. In customer Bridges the service instance is identified by a C-VID, and in Provider Bridges by an S-VID.

Insert the following definitions, and renumber the definitions in alphabetical order.

3.1 Active SAP: The SAP, bounding an MP, that is on the side of the MP towards the monitored MA.

3.2 Connectivity Fault Management (CFM): Connectivity Fault Management comprises capabilities for detecting, verifying, and isolating connectivity failures in Virtual Bridged Local Area Networks.

3.3 Continuity Check Message (CCM): A multicast CFM PDU transmitted periodically by a MEP in order to ensure continuity over the MA to which the transmitting MEP belongs. No reply is sent by any MP in response to receiving a CCM.

3.4 Domain Service Access Point (DoSAP): A member of a set of SAPs at which a Maintenance Domain is capable of offering connectivity to systems outside the Maintenance Domain. Each DoSAP provides access to an instance either of the EISS or of the ISS.

3.5 Down MEP: A MEP residing in a Bridge that receives CFM PDUs from, and transmits them towards, the direction of the LAN.

3.6 Down MP: A MEP or an MHF residing in a Bridge that receives CFM PDUs from, and transmits them towards, the direction of the LAN.

3.7 EISS Service Access Point (EISS-SAP): An instance of the EISS.

NOTE—See 6.8.

3.8 Fault Alarm: An out-of-band signal, which can be an SNMP Notification, that notifies a system administrator of a connectivity failure.

3.9 Intermediate Service Access Points (ISAP): A SAP, interior to a Maintenance Domain, through which frames can pass in transit from DoSAP to DoSAP.

3.10 ISS Service Access Point (ISS-SAP): An instance of the ISS.

NOTE—See 6.6.

3.11 Layer Management Interface: An interface used for carrying management controls, counters, status parameters, or event indications that are not communicated to an entity's peers.

NOTE—This is not one of the SAPs defined in this standard.

3.12 Linktrace Message (LTM): A CFM PDU initiated by a MEP to trace a path to a target MAC address, forwarded from MIP to MIP, up to the point at which the LTM reaches its target, a MEP, or can no longer be forwarded. Each MP along the path to the target generates an LTR.

3.13 Linktrace Output Multiplexer (LOM): A CFM shim that enables the Linktrace Responder to forward LTMs out a specific Bridge Port without passing the LTMs through the Frame filtering entity.

3.14 Linktrace Reply (LTR): A unicast CFM PDU sent by an MP to a MEP, in response to receiving an LTM from that MEP.

3.15 Loopback Message (LBM): A unicast CFM PDU transmitted by a MEP, addressed to a specific MP, in the expectation of receiving an LBR.

3.16 Loopback Reply (LBR): A unicast CFM PDU transmitted by an MP to a MEP, in response to an LBM received from that MEP.

3.17 Maintenance Association (MA): A set of MEPs, each configured with the same MAID and MD Level, established to verify the integrity of a single service instance. An MA can also be thought of as a full mesh of Maintenance Entities among a set of MEPs so configured.

3.18 Maintenance association End Point (MEP): An actively managed CFM entity, associated with a specific DoSAP of a service instance, which can generate and receive CFM PDUs and track any responses. It is an end point of a single MA and is an end point of a separate Maintenance Entity for each of the other MEPs in the same MA.

3.19 Maintenance association End Point Identifier (MEPID): A small integer, unique over a given MA, identifying a specific MEP.

3.20 Maintenance Association Identifier (MAID): An identifier for a Maintenance Association, unique over the domain that CFM is to protect against the accidental concatenation of service instances. The MAID has two parts: the Maintenance Domain Name and the Short MA Name.

3.21 Maintenance C-VLAN: One C-VLAN, among a number of C-VLANs carried over a single service instance, that is used for CFM PDUs.

3.22 Maintenance Domain: The network or the part of the network for which faults in connectivity can be managed. The boundary of a Maintenance Domain is defined by a set of DoSAPs, each of which can become a point of connectivity to a service instance.

3.23 Maintenance domain Intermediate Point (MIP): A CFM entity consisting of two MHFs.

3.24 Maintenance Domain Name: The identifier, unique over the domain for which CFM is to protect against accidental concatenation of service instances, of a particular Maintenance Domain.

3.25 Maintenance Entity (ME): A point-to-point relationship between two MEPs within a single MA.

3.26 Maintenance Point (MP): One of either a MEP or a MIP.

3.27 MD Level: A small integer in a field in a CFM PDU that is used, along with the VID in the VLAN tag, to identify to which Maintenance Domain among those associated with the CFM frame’s VID, and thus to which MA, a CFM PDU belongs. The MD Level determines the MPs that are a) interested in the contents of a CFM PDU, and b) through which the frame carrying that CFM PDU is allowed to pass.

3.28 MEP CCM Database: A database, maintained by every MEP, that maintains received information about other MEPs in the Maintenance Association.

3.29 MIP CCM Database: A database, maintained optionally by a MIP or MEP, that maintains received information about the MEPs in the Maintenance Domain.

3.30 MIP Half Function (MHF): A CFM entity, associated with a single Maintenance Domain, and thus with a single MD Level and a set of VIDs, that can generate CFM PDUs, but only in response to received CFM PDUs.

3.31 MP Level Demultiplexer: The component of an MP that separates CFM PDUs at different MD Levels.

3.32 MP Multiplexer: A component of an MP that merges frames from more than one input SAP, sending them to a single output SAP.

3.33 MP Type Demultiplexer: A component of an MP that identifies and separates CFM PDUs based on the Type/Length field.

3.34 MP OpCode Demultiplexer: A component of an MP that identifies and separates CFM PDUs with different values of the OpCode field.

3.35 Operations, Administration, and Maintenance (OAM): A group of network management functions that provide network fault indication, performance information, and data and diagnosis functions. (adapted from ATM Forum Glossary)

NOTE 1—Examples are ATM OAM {ITU-T I.610 (1999) [B13]}[3] and IEEE 802.3 Clause 57 OAM.

3.36 operator: An operator provides a single network of Provider Bridges or a single Layer 2 or Layer 3 backbone network to the service provider.

NOTE 1—An operator can, in fact, be identical to, or a part of the same organization as, the service provider, but for the purposes of this standard, the operator and service provider are presumed to be separate organizations.

NOTE 2—Certain terms in this standard, including "customer," "service provider," and "operator" reflect common business relationships that often exist among organizations and individuals that use equipment implemented in accordance with this standard. Terms such as "customer VID" or "operator MD Level" are no more than convenient labels for entities defined here; their use does not imply any requirement upon the business relationships among vendors or users of devices compliant with this standard.

3.37 Owner: An Owner of a system (and in particular, a Bridge) is a user who has full access to the System Group of the SNMPv2-MIB (IETF RFC 3418).

3.38 Passive SAP: The SAP, bounding an MP, that is on the side of the MP away from the monitored MA.

3.39 Primary VID: The VID, among a list of VIDs associated with a service instance, on which all CFM PDUs generated by MPs except for forwarded LTMs are to be transmitted.

3.40 Service Access Point (SAP): The point at which a service is offered.

NOTE—In this standard, an instance of either the ISS (6.6) or the EISS (6.8). This term is used in place of the awkward "ISS Service Access Point" or "EISS Service Access Point," where no loss of clarity is introduced.

3.41 shim: A protocol entity that uses the same service as it provides.

NOTE—Within this standard, shims make use of the ISS or the EISS.

[3]The numbers in brackets correspond to those of the bibliography in Annex H.

3.42 Short MA Name: The part of the MAID that is unique within the Maintenance Domain and is appended to the Maintenance Domain Name to form the MAID.

3.43 Up MEP: A MEP residing in a Bridge that transmits CFM PDUs towards, and receives them from, the direction of the Bridge Relay Entity.

3.44 Up MP: A MEP or an MHF residing in a Bridge that transmits CFM PDUs towards, and receives them from, the direction of the Bridge Relay Entity.

4. Abbreviations

Insert the following abbreviations in alphabetical order:

CCM	Continuity Check Message
CFM	Connectivity Fault Management
DoSAP	Domain Service Access Point
ISAP	Intermediate Service Access Point
ITU-T	International Telecommunications Union—Telecommunication Standardization Sector
LBM	Loopback Message
LBR	Loopback Reply
LMI	Layer Management Interface
LOM	Linktrace Output Multiplexer
LTM	Linktrace Message
LTR	Linktrace Reply
ME	Maintenance Entity
MA	Maintenance Association
MAID	Maintenance Association Identifier
MEP	Maintenance association End Point
MEPID	Maintenance association End Point Identifier
MHF	MIP Half Function
MIP	Maintenance domain Intermediate Point
MP	Maintenance Point
MSAP	MAC Service Access Point

5. Conformance

5.4.1 VLAN-aware Bridge component options

Insert the following item after item b) in 5.4.1 and renumber the list items:

c) Support CFM operation (5.4.1.3);

Insert the following subclause after 5.3.1.2:

5.4.1.3 Connectivity Fault Management (optional)

A VLAN-aware Bridge component implementation that conforms to the provisions of this standard for Connectivity Fault Management (Clause 18, Clause 19, Clause 20, Clause 21, Clause 22) shall:

a) Support the creation of Maintenance Domains at eight MD Levels, with multiple non-overlapping Maintenance Domains at each MD Level, using the managed objects specified in 12.14;
b) Support the creation of a Maintenance Association (MA) on each VLAN supported by the Bridge for each MD Level;
c) Support the creation of a single MIP for each Maintenance Domain on each Port, all MIPs being at the same MD Level;
d) Support the creation of eight Up MEPs on each VLAN on each Port, each MEP at a different MD Level;
e) Support the creation of eight Down MEPs on each VLAN on each Port, each MEP at a different MD Level;
f) Support the creation of eight Down MEPs attached to no VLAN on each Port, each MEP at a different MD Level;
g) Support the maintenance of a MEP CCM Database;
h) Provide control via all of the required managed objects specified in 12.14;
i) Conform to the state machines and procedures in Clause 20; and
j) Transmit and receive required CFM PDUs in the formats specified in Clause 21.

A VLAN-aware Bridge component implementation that conforms to the provisions of this standard for Connectivity Fault Management may:

k) Support the creation of MIPs at different MD Levels on a single Port;
l) Support the creation of MIPs at lower or equal MD Levels than MEPs on the same Port;
m) Support the maintenance of a MIP CCM Database in a MIP or MEP;
n) Support the CFM Management Information Base (MIB) module defined in 17.5; and/or
o) Support the creation of MAs that are associated with more than one VLAN (see 22.1.5).

Insert the following subclause after 5.9:

5.10 VLAN-aware end station requirements for Connectivity Fault Management

A VLAN-aware Station implementation that conforms to the provisions of this standard for Connectivity Fault Management (Clause 18, Clause 19, Clause 20, Clause 21, Clause 22) shall:

a) Support the creation of Maintenance Domains at eight MD Levels, with multiple non-overlapping Maintenance Domains at each MD Level, using the managed objects specified in 12.14;
b) Support the creation of a Maintenance Association (MA) on each VLAN supported by the Station for each MD Level;
c) Support the creation of eight Down MEPs on each VLAN on each Port, each MEP at a different MD Level;

d) Support the creation of eight Down MEPs attached to no VLAN on each Port, each MEP at a different MD Level;
e) Support the maintenance of a MEP CCM Database;
f) Provide control via all of the required managed objects specified in 12.14;
g) Conform to the state machines and procedures in Clause 20; and
h) Transmit and receive required CFM PDUs in the formats specified in Clause 21.

A VLAN-aware Station implementation that conforms to the provisions of this standard for Connectivity Fault Management may:

i) Support the CFM Management Information Base (MIB) module defined in 17.5; and/or
j) Support the creation of MAs that are associated with more than one VLAN (see 22.1.5).

6. Support of the MAC Service ~~in VLANs~~

Change the title of Clause 6, and delete the last paragraph of the introductory text as shown:

~~The provisions of IEEE Std 802.1D, Clause 6, apply to this standard, with the additions and modification defined in this clause.~~

Insert the following text after the introductory text to Clause 6, before 6.1:

This clause:

a) Summarizes basic architectural concepts and terms used throughout this standard, introduces the primitives and parameters of the MAC Service, and defines VLANs in terms of the connectivity provided to service users (6.1, 6.2, 6.3);
b) Describes how Bridges preserve and maintain the quality of the MAC Service (6.4, 6.5);
c) Specifies the MAC Internal Sublayer Service (ISS) and MAC status parameters, and their support by specific media access methods (6.6, 6.7, 6.11);
d) Specifies the Enhanced Internal Sublayer Service (EISS) used by VLAN-aware stations, the encoding of VLAN identifier and priority parameters in transmitted frames, and the classification of received frames that do not explicitly convey those parameters (6.8, 6.9, 6.10); and
e) Specifies how entities defined in terms of the ISS can support the EISS (6.15).

Insert the following new subclauses before 6.1, and renumber the subsequent subclauses.

6.1 Basic architectural concepts and terms

The architectural concepts used in this and other IEEE 802.1 standards are based on the layered protocol model introduced by the OSI Reference Model (ISO/IEC 7498-1) and used in the MAC Service Definition (ISO/IEC 15802-1), in IEEE Std 802®, in other IEEE 802 standards, and elsewhere in networking. IEEE 802.1 standards in particular have developed terms and distinctions useful in describing the MAC Service and its support by protocol entities within the MAC Sublayer.[4]

6.1.1 Protocol entities, peers, layers, services, and clients

The fundamental notion of the model is that each protocol entity within a system is instantiated at one of a number of strictly ordered layers, and communicates with peer entities (operating the same or an interoperable protocol within the same layer) in other systems by using the service provided by interoperable protocol entities within the layer immediately below, and thus provides service to protocol entities in the layer above. The implied repetitive stacking of protocol entities is essentially unbounded at the lowest level and is bounded at the highest level by an application supported by peer systems. In descriptions of the model, the relative layer positions of protocol entities and services is conventionally referred to by N, designating a numeric level. The N-service is provided by an N-entity that uses the (N – 1) service provided by the (N – 1) entity, while the N-service user is an (N + 1) entity. Figure 6-1 illustrates these concepts with reference to the MAC Sublayer, which contains MAC entities that provide the MAC Service, to MAC Service users.

6.1.2 Service interface primitives, parameters, and frames

Each N-service is described in terms of service primitives and their parameters, each primitive corresponding to an atomic interaction between the N-service user and the N-service provider, with each invocation of a primitive by a service user resulting in the service issuing corresponding primitives to peer

[4]Drawing on prior network layer standards, including ISO/IEC 7498-1, wherever possible.

service users. The purpose of the model is to provide a framework and requirements for the design of protocols while not unnecessarily constraining the internal design of systems: primitives and their parameters are limited to, but include all of the information elements conveyed to corresponding peer protocol entities or required by other systems (and not supplied by protocols in lower layers) to identify (address) those entities and deliver the information. The parameters of service primitives do not include information that is used only locally, i.e., within the same system, to identify entities or organize resources for example.[5]

The primitives of the MAC Service comprise a data request and a corresponding data indication, each with MAC destination address, MAC source address, a MAC service data unit comprising one or more octets of data, and priority parameters. Taken together these parameters are conveniently referred to as a frame, although this does not imply that they are physically encoded by a continuous signal on a communication medium, that no other fields are added or inserted by other protocol entities prior to transmission, or that the priority is always encoded with the other parameters transmitted.

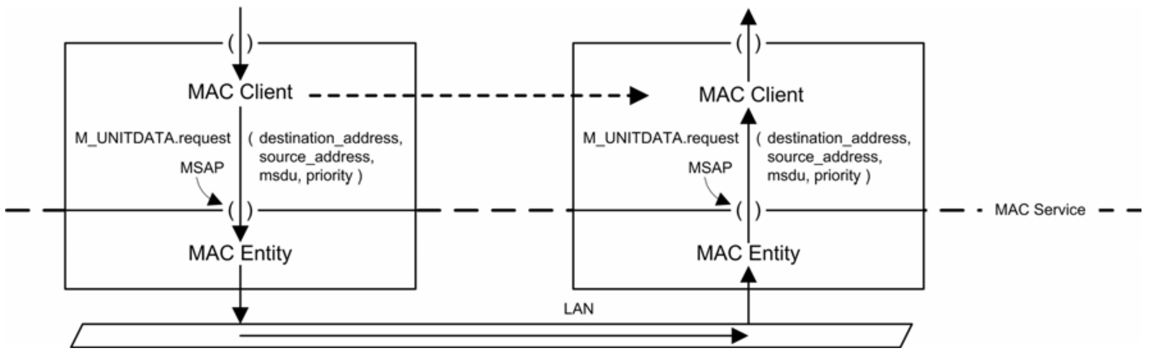
MAC Client
M_UNITDATA.request
(destination_address, source_address, msdu, priority)
MSAP
MAC Entity
MAC Client
M_UNITDATA.request
(destination_address, source_address, msdu, priority)
MSAP
MAC Entity
MAC Service
LAN

Figure 6-1—MAC entities, the MAC Service, and MAC Service users (clients)

6.1.3 Layer management interfaces

A given N-entity can have many associated management controls, counters, and status parameters that are not communicated to its user's peers and whose values are either not determined by its user or not required to change synchronously with the occurrence of individual N-service primitives to ensure successful (N + 1)-protocol operation. Communication of the values of these parameters to and from local entities, i.e., within the same system, is modeled as occurring not through service primitives[6] but through a layer management interface (LMI). One protocol entity, for example an SNMP entity, can be used to establish the operational parameters of another. Communication of the results of Fault Alarms to entities responsible for managing the network is one of the uses of LMIs in this standard.

[5]These points are frequently misunderstood by those unfamiliar with the reference model, who take it as simply restating common sense principles of modular engineering. Early variants of the MAC Service, for example, omitted the source MAC address parameter on the grounds that it was a fixed property of the transmitting station and should be supplied by the MAC entity itself, despite the fact that communicating peer service users (and the protocols they operate) required that information. The introduction of MAC Bridges necessitated the development of a MAC Internal Sublayer Service with the required parameter, and has led to a restatement of the service definition included in a number of standards. However the source address parameter would still have been required even if MAC Bridges did not exist. Similarly versions of the MAC Service have included local acknowledgment primitives or status return codes for primitives issued. These play no part in defining the peer-to-peer communication and do not conform to the reference model. The scope of some IEEE 802 standards does include the definition of interfaces, particularly electrical interfaces, within systems. These play a valuable role in defining components used to build those systems, but should not be confused with OSI service interfaces.

[6]This would require considerable enlargement and continuous modification of service interfaces, obscuring their original purpose, not to mention the creation of many additional interfaces and the addition of "pass-through" functions to others.

6.1.4 Service access points, interface stacks, and ports

Each service is provided to a single protocol entity at a service access point (SAP) within a system. A given N-entity can support a number of N-SAPs and use one or more (N – 1)-SAPs. The SAP serves to delineate the boundary between protocol specifications and to specify the externally observable relationship between entities operating those protocols. An SAP is an abstraction and does not necessarily correspond to any concrete realization within a system, but an N-entity often associates management counters with the SAP and provides status parameters that can be used by the (N + 1)-entity using the SAP. Examples include the MAC_Operational and operPointToPoint MAC status parameters (6.6.2, 6.6.3).

The network and link layers[7] of the reference model accommodate many different real networks, subnetworks, and links with the requirements for bandwidth, multiplexing, security, and other aspects of communication differing from network to network. A given service, e.g., the MAC Service, is often provided by a number of protocols, layered to achieve the desired result. Together the entities that support a particular SAP compose an interface stack. Figure 6-2 provides an example, showing the use of Link Aggregation (IEEE Std 802.3, Clause 43).

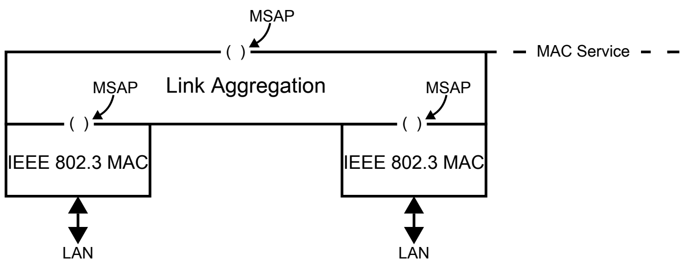

Figure 6-2—An interface stack

The term “port” is used to refer to the interface stack for a given SAP. Often the interface stack comprises a single protocol entity attached to a single LAN, and port can be conveniently used to refer to several aspects of the interface stack, including the physical interface connector for example. In more complex situations—such as that in Figure 6-2, where the parts of the interface stack provided by the IEEE 802.3 MAC entities effectively compose two ports that are then used by link aggregation to provide a single port to its user—the port has to be clearly specified in terms of the particular SAP supported.

6.1.5 MAC method independent protocols and shims

Protocols specified in IEEE Std 802.3, IEEE Std 802.11™-2007 [B5], and other IEEE 802 standards, can be specific to their LAN media or to the way access to that media is controlled. Other protocols and functions within the MAC sublayer, such as link aggregation and bridging, are MAC method independent—thus providing consistent management and interoperability across a range of media.

Definition of a service facilitates the specification of shims, i.e., protocol entities that use the same service as they provide. Protocol shims can be inserted into an interface stack to provide aggregation (e.g., IEEE Std 802.3, Clause 43), security (e.g., IEEE Std 802.1AE™-2006 [B4]), or multiplexing.

[7]The data link layer, as originally envisaged by the OSI reference model, contained no addressing and caused some involved in its development to reject the idea of LANs at the link layer. There is a sound argument for regarding LANs as simply subnetworks within the network layer, and in practice this is how they are treated. However this would have been unpalatable to many more people at a time when correspondence between LLC (IEEE Std 802.2) and HDLC was sought, together with the adoption of a unique network layer protocol (X.25). Continuing to regard the MAC Sublayer as part of the OSI Data Link Layer does relatively little harm, except when duplication of network layer functionality is proposed, and is convenient given the mass of historic documentation.

6.1.6 MAC Service clients

The protocol entity that uses the service provided at an MSAP is commonly referred to as client of the MAC Service or of the entity providing the service. Clients of the MAC Service (or ISS) include the MAC Relay Entity and the LLC Entity specified in IEEE Std 802.2. The latter provides protocol identification, multiplexing and demultiplexing to and from a number of clients that use a common MSAP. The clients of LLC are also often referred to as clients of the MAC.

NOTE—For the purposes of this standard, the terms "LLC" and "LLC Entity" include the service provided by the operation of entities that support protocol discrimination using an Ethertype, i.e., protocol discrimination based on the Type interpretation of the Ethertype field as specified in IEEE Std 802a™-2003 [B2].

6.1.7 Stations and systems

A LAN station comprises a single media access method specific entity, operating the MAC procedures specified in the applicable IEEE 802 standard, together with other protocol entities mandated by those standards (e.g., an LLC Entity) or commonly used in conjunction with that entity.

A system is a collection of hardware and software components whose intercommunication is not directly externally observable and outside the scope of the IEEE 802 standards that specify the system behavior as a whole. Management of a system, when supported, is typically provided through a single management entity. A system (such as a bridge) can contain many media access method specific entities, of the same or a variety of types, attached to different LANs. A system can therefore be said to comprise one or more LAN stations.

6.1.8 Connectionless connectivity

The MAC Service supported by an IEEE 802 LAN provides connectionless connectivity, i.e., communication between attached stations occurs without explicit a priori agreement between service users. The potential connectivity offered by a connectionless service composes a connectivity association that is established prior to the exchange of service primitives between service users. The way in which such a connectivity association is established depends on the particular protocols and resources that support it, and can be as simple as making a physical attachment to a wire. However simple or complex, the establishment of a connectivity association for connectionless data transfer involves only a two-party interaction between the service user and the service provider (though it can result in exchanges between service providing entities in several systems) and not a three-party user-service-user interaction as is the case for connection-oriented communication. With the continual increase in the number of ways that IEEE 802 LAN connectivity can be supported it is no longer useful to regard a LAN as a definite set of physical equipment, instead the connectivity association that exists between a set of MAC Service access points defines a LAN.[8]

6.2 Provision of the MAC service

The MAC Service provided in end stations attached to a Bridge Local Area Networks and Virtual Bridged Local Area Network is the (unconfirmed) connectionless mode MAC Service defined in ISO/IEC 15802-1. The MAC Service is defined as an abstraction of the features common to a number of specific MAC Services; it describes the transfer of user data between source and destination end stations, via MA-UNITDATA request primitives and corresponding MA-UNITDATA indication primitives issued at MAC Service access points. Each MA-UNITDATA request and indication primitive has four parameters: Destination Address, Source Address, MAC Service data unit (MSDU), and Priority.

NOTE 1—The primitives of the MAC Service are closely aligned with those of the ISS (6.6).

[8]A LAN is thus defined in terms of its external observable behavior, not by an abstraction of its internal operation.

In a Bridged Local Area Network, MAC Bridges (IEEE Std 802.1D) provide a single instance of the MAC Service for the network as a whole—a single LAN, as previously defined (6.1.8)—by forwarding between the individual LANs that compose the network, though frames with specified group destination addresses are confined to individual LANs.

NOTE 2—MAC Bridges can be configured to completely partition a bridged network, though that is rarely intended.

In a Virtual Bridged Local Area Network, Virtual LANs are defined in terms of the connectivity associations (6.1.8) supported by the network: if service requests made at two different MSAPs both result in service indications at any third MSAP, then the two MSAPs are transmitting on the same VLAN; and if the service requests made at any given MSAP can result in service indications at either (or both) of two MSAPs, then those two MSAPs are receiving on the same VLAN. When only two MSAPs are in bidirectional communication, they are using the same VLAN for both transmit and receive if a third station could be added in such a way that it would receive transmissions from both.

NOTE 3—This definition of a VLAN accommodates an unavoidable and sometimes useful consequence of configuration possibilities: use of a single MSAP can result in transmission on one VLAN and reception on another.

NOTE 4—Protocol VLAN classification can be used to classify frames transmitted from a single MSAP into separate VLANs. Where protocol classification is used, the preceding rules apply to frames of a single classification.

6.3 Support of the MAC service

Delete the first paragraph in 6.3 as follows.

~~The MAC Service (MS) provided to end stations attached to a Virtual Bridged Local Area Network is the (unconfirmed) connectionless mode MAC Service defined in ISO/IEC 15802-1. The MAC Service is defined as an abstraction of the features common to a number of specific MAC Services; it describes the transfer of user data between source and destination end stations, via MA-UNITDATA request primitives and corresponding MA-UNITDATA indication primitives issued at MAC Service access points. Each MA-UNITDATA request and indication primitive has four parameters: Destination Address, Source Address, MAC Service data unit (MSDU), and Priority.~~

Delete the last paragraph of 6.3 and the accompanying NOTE as follows:

~~The operation of Bridges supports the provision of the MAC Service only to devices that are authenticated and authorized for such use. Unauthorized devices may be denied access to the network, other than as necessary to support the protocol exchanges that are required by any authentication process that is supported.~~

~~NOTE—Authentication and authorization to access a LAN may be achieved by administrative or management mechanisms, or by means of an active authorization mechanism, such as is defined in IEEE Std 802.1X-2001.~~

Insert the following text at the beginning of 6.3:

In a Bridged Local Area Network at most one LAN can be supported on each of the individual LANs that, bridged together, compose the network. On the individual LANs of a Virtual Bridged Local Area Network, frames for different VLANs can be distinguished by the addition of a VLAN tag as the initial octets of a frame's MSDU.

When a VLAN-unaware end station is attached to an individual LAN, it will transmit and receive only untagged frames, and thus uses the VLAN(s) assigned by the Bridge Ports that classify those frames.

NOTE 1—Where there are only two stations attached to an individual LAN, each can assign received untagged frames to a different VLAN. However such assignments can defeat the operation of network configuration protocols and can only be recommended at the edge of a network to facilitate initial classification of untagged frames transmitted by VLAN-unaware end stations. Otherwise a change in a frame's VLAN can severely impact network operation by introducing frame forwarding loops. The same considerations apply to the use of protocol VLAN classification. The use of different VLAN assignments by Bridge Ports attached to the same LAN is usually a network configuration error.

NOTE 2—The LLC Entity in a VLAN-unaware end station can receive MAC Service indications that correspond to the receipt of VLAN-tagged frames but, being unable to recognize the tag's Ethertype value, will discard the frame and not deliver it to any LLC User.

A VLAN-aware end station uses the EISS Multiplex Entity (6.15) to provide multiple MSAPs, one per VLAN of interest, to separate MAC Clients.

Insert a new subclause at the end of Clause 6:

6.15 EISS Multiplex Entity

The EISS Multiplex Entity enables shims defined for the ISS to use the EISS. Figure 6-3 illustrates two EISS Multiplex Entities placed back-to-back.

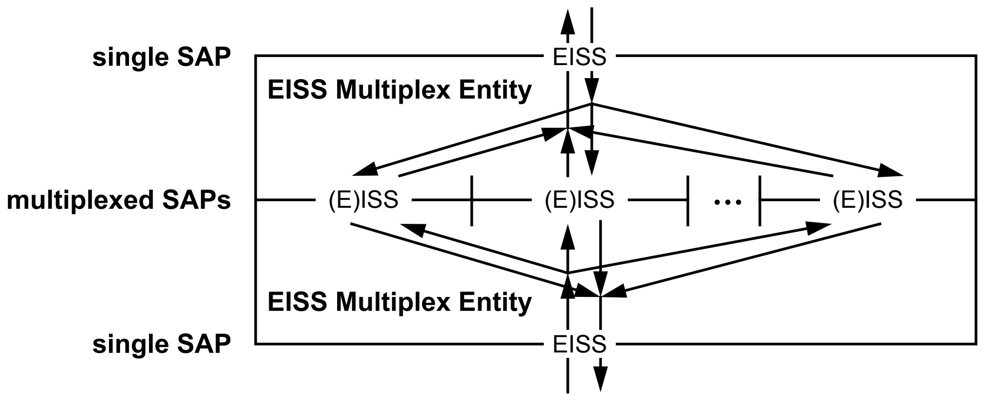

Figure 6-3—Two back-to-back EISS Multiplex Entities

An EISS Multiplex Entity has one EISS SAP, and a number of multiplexed SAPs, each either an ISS SAP or an EISS SAP. Each multiplexed ISS SAP is assigned to a single vlan_identifier value. Each multiplexed EISS SAP is assigned to one or more vlan_identifier values. Every vlan_identifier is assigned to some multiplexed SAP. Upon receiving a Request or Indication from its single EISS SAP, the EISS Multiplex Entity uses the vlan_identifier and canonical_format_indicator to select the corresponding one of its multiplexed SAPs to present the Request or Indication. Similarly, any Request or Indication received from a multiplexed SAP is presented to the single EISS SAP; the vlan_identifier and canonical_format_indicator parameters presented on the single EISS SAP are the ones associated with the multiplexed ISS SAP, or the ones obtained from the multiplexed EISS SAP.

Shims can be multiplied in this fashion to separately serve multiple vlan_identifiers. Figure 22-1 illustrates CFM shims deployed in this manner.

The rif_information parameter is not a parameter of the single EISS of an EISS Multiplex Entity if it uses the ISS on any of its multiplexed SAPs. The ISS does not support media that require the rif_information parameter.

The MAC_Operational status parameter (6.6.2) presented to the uppermost EISS-SAP in Figure 6-3 is true if and only if:

a) The uppermost EISS-SAP’s MAC_Enabled parameter is true; and
b) At least one of its multiplexed SAPs’ MAC_Operational status parameters is true.

The MAC_Operational status parameter of each of the lower EISS Multiplex Entity’s multiplexed SAPs in Figure 6-3 is computed separately and is true if and only if:

c) The lowermost EISS-SAP’s MAC_Operational parameter is true; and
d) That multiplexed SAP’s MAC_Enabled parameter is true.

8. Principles of bridge operation

8.3 Model of operation

Insert item 4) at the end of list b):

4) Linktrace response

Insert the following list immediately following list b):

c) Zero or more Connectivity Fault Management (CFM) Entities, each of which can be either:
   1) A Maintenance association End Point (MEP);
   2) A Maintenance association Intermediate Point (MIP); or
   3) A Linktrace Output Multiplexer.

8.6.7 Queue management

Change item c) and insert a NOTE immediately following item c) as follows:

c) If the associated Port leaves the Forwarding state. Frames entering the queue subsequent to the transition out of the Forwarding state (e.g., CFM frames from Down MPs, as described in 22.1, Figure 22-4, and Figure 22-7) are not discarded.

NOTE—CFM frames may be excepted, and not removed from the queue, on a transition out of the Forwarding state (see 22.1.3).

8.12 Bridge management entity

Change the following list items, delete footnotes 16–18, and renumber the remaining footnotes:

b) ~~IETF RFC 1493~~IETF RFC 4188;
c) ~~RSTP MIB~~IETF RFC 4318;
d) ~~MSTP MIB.~~IETF RFC 4363;

Insert item e) immediately following item d):

e) CFM MIB (17.5).

8.13 Addressing

Insert the following subclause after 8.13.10.

8.13.11 Connectivity Fault Management entities

CFM entities reside in shims. The entities in a CFM shim inspect every frame that passes through either of the shim’s two SAPs, and decides whether to pass that frame through to the other SAP, process it, or both. The decision whether pass it through is not dependent on the destination_address, but the decision whether to process the frame is dependent on the destination_address (see Clause 19). A CFM entity shall discard any frame, directed to it by reason of its VID and MD Level, whose destination_address parameter does not correspond to a MAC address recognized by that CFM entity.

Any given CFM entity is configured to recognize one Individual MAC address, and one or more Group MAC addresses, in received frames, and to use one or two of the Group MAC addresses for transmitted frames. The Individual MAC address is unique, within a bridged network, to a specific Bridge, though not necessarily to a single Bridge Port (J.6). The Group MAC addresses are selected from Table 8-9 and Table 8-10 according to the MD Level and OpCode fields in the CFM PDU header. Loopback Messages (LBMs, 20.2), Loopback Replies (LBRs, 20.2), and Linktrace Replies (LTRs, 20.3) are carried in unicast frames. Continuity Check Messages (CCMs, 20.1) use addresses from Table 8-9, and Linktrace Messages (LTMs, 20.3) use addresses from Table 8-10.

Table 8-9—Continuity Check Message Group Destination MAC Addresses

01-80-C2-00-00-3y	
MD Level of CCM	**Four address bits "y"**
7	7
6	6
5	5
4	4
3	3
2	2
1	1
0	0

Table 8-10—Linktrace Message Group Destination MAC Addresses

01-80-C2-00-00-3y	
MD Level of LTM	**Four address bits "y"**
7	F
6	E
5	D
4	C
3	B
2	A
1	9
0	8

NOTE—The Group destination MAC addresses in Table 8-9 are there primarily to make it possible for Bridges built prior to this standard to process CFM. In future revisions of this standard, it is possible that this restriction on the choice of Group destination MAC addresses to be used on a CFM PDU will be relaxed. See 22.8.

12. Bridge management

12.1 Management functions

Insert the following paragraph at the end of 12.1.

It can be necessary, particularly in Provider Bridged Networks, for multiple organizations to manage a single Bridge, with each organization having different abilities to manage, or even detect the existence of, Bridge Ports, Bridges, or entire networks. Access to all of the managed objects in Clause 12 for reading, writing, or detection can be restricted. Access by certain organizations can be limited to only a small number of managed objects, particularly to those specified in 12.14.

12.1.1 Configuration management

Insert the following item g) after the last list item.

g) The ability to create and delete the functional elements of Connectivity Fault Management and to control their operation.

12.1.2 Fault management

Insert the following item h) after the last list item.

h) Within the context of multiple management organizations, the ability to:
   1) Actively monitor the connectivity of a set of managed end points on individual VLANs, simultaneously over multiple physical extents;
   2) Actively monitor and ensure the segregation of data among different VLANs;
   3) Issue trains of point-to-point query-response messages to selected network components in a VLAN; and
   4) Issue multicast query-relay-response messages to determine the path taken by frames addressed to specific individual MAC addresses through a VLAN.

12.2 Managed objects

Insert the following item i) after the last list item.

i) The CFM Entities (12.14).

Insert the following subclause at the end of Clause 12.

12.14 CFM entities

The CFM managed objects model operations that modify, or enquire about, the configuration and the operation of CFM. Figure 12-1 illustrates the simple hierarchical relationships among the various CFM managed objects, including the control of access to those objects. See 17.5 for an explanation of methods available to distinguish between an Owner and a Maintenance Domain administrator. See Clause 18 for an explanation of the use of Maintenance Domains and Maintenance Associations (MAs).The following are the CFM managed objects in a Bridge:

a) Maintenance Domain list managed object (12.14.1);
b) CFM Stack managed object (12.14.2);
c) Default MD Level managed object (12.14.3);
d) Configuration Error List managed object (12.14.4);
e) Maintenance Domain managed objects (12.14.5);

f) Maintenance Association managed objects (12.14.6); and
g) Maintenance association End Point managed objects (12.14.7).

12.14.1 Maintenance Domain list managed object

There is one Maintenance Domain list managed object per Bridge. It contains a list of the Maintenance Domains that have been configured on the Bridge.

The management operations that can be performed on the Maintenance Domain list managed object are as follows:

a) Read Maintenance Domain list (12.14.1.1);
b) Create Maintenance Domain managed object (12.14.1.2); and
c) Delete Maintenance Domain managed object (12.14.1.3).

NOTE—In Provider Bridge applications, each Maintenance Domain, and thus each Maintenance Domain managed object, can be controlled by a different administrative organization. Some or all of these administrative organizations can be different from that which controls the Maintenance Domain list managed object or the managed objects in Clause 12 other than those in 12.14. See 17.4 for a discussion of the means by which control of and access to different instances of the Maintenance Domain managed object are allocated to these different administrative organizations.

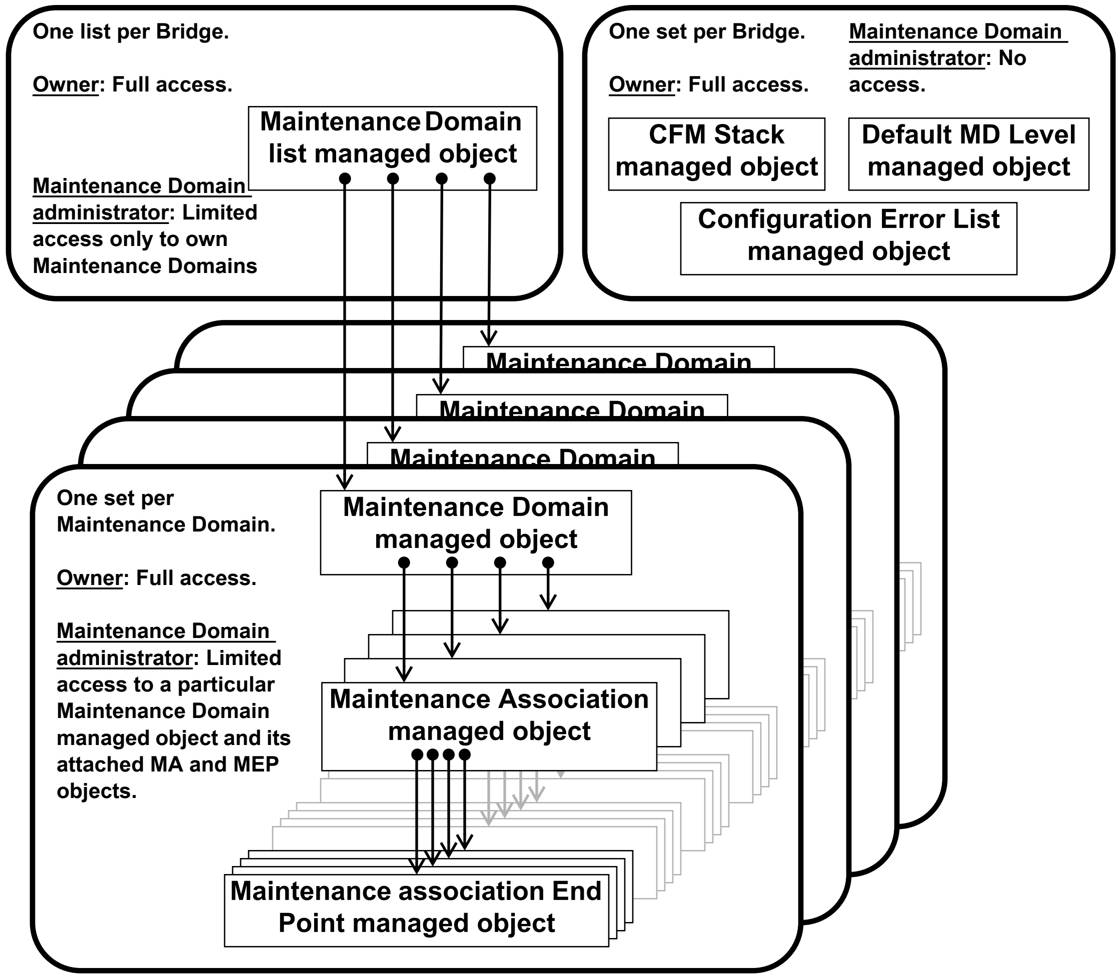
One list per Bridge.
Owner: Full access.
Maintenance Domain list managed object
Maintenance Domain administrator: Limited access only to own Maintenance Domains
One set per Bridge.
Owner: Full access.
Maintenance Domain administrator: No access.
CFM Stack managed object
Default MD Level managed object
Configuration Error List managed object
Maintenance Domain
Maintenance Domain
Maintenance Domain
One set per Maintenance Domain.
Owner: Full access.
Maintenance Domain administrator: Limited access to a particular Maintenance Domain managed object and its attached MA and MEP objects.
Maintenance Domain managed object
Maintenance Association managed object
Maintenance association End Point managed object

Figure 12-1—Relationships among CFM managed objects

12.14.1.1 Read Maintenance Domain list

12.14.1.1.1 Purpose

To obtain information about all of the Maintenance Domains in a Bridge.

12.14.1.1.2 Inputs

None.

12.14.1.1.3 Outputs

a) A list, perhaps empty, of the Maintenance Domain managed objects configured on the Bridge. For each item in the list, the Read Maintenance Domain list command returns:
   1) A Maintenance Domain Name, including the format specifier from Table 21-19 (21.6.5.1); and
   2) A reference to that Maintenance Domain's Maintenance Domain managed object (12.14.5).

12.14.1.2 Create Maintenance Domain managed object

12.14.1.2.1 Purpose

To create a new Maintenance Domain managed object in a Bridge, and add it to the Bridge's Maintenance Domain list managed object.

12.14.1.2.2 Inputs

a) A Maintenance Domain Name, including the format specifier from Table 21-19 (21.6.5.1); default value is the character string (format 4) "DEFAULT"; and
b) The MD Level at which the Maintenance Domain is to be created; default value is 0.

12.14.1.2.3 Outputs

a) Operation status. This takes one of the following values:
   1) Operation rejected due to invalid Maintenance Domain Name (21.6.5.1);
   2) Operation rejected because Maintenance Domain Name already exists;
   3) Operation rejected due to invalid MD Level; or
   4) Operation accepted.

12.14.1.3 Delete Maintenance Domain managed object

12.14.1.3.1 Purpose

To delete a specific Maintenance Domain managed object from a Bridge and from the Bridge's Maintenance Domain list managed object.

12.14.1.3.2 Inputs

a) A reference to a particular Maintenance Domain managed object (12.14.5).

12.14.1.3.3 Outputs

a) Operation status. This takes one of the following values:
   1) Operation rejected due to invalid or nonexistent Maintenance Domain; or
   2) Operation accepted.

12.14.2 CFM Stack managed object

There is one CFM Stack managed object per Bridge. It permits retrieval of information about the Maintenance Points configured on a particular Bridge Port or aggregated port. The management operation that can be performed is:

a) Read CFM Stack managed object (12.14.2.1).

12.14.2.1 Read CFM Stack managed object

12.14.2.1.1 Purpose

To obtain the contents of the CFM Stack managed object, which allows a network administrator to discover the relationships among MEPs and MHFs configured on a particular Bridge Port or aggregated port.

12.14.2.1.2 Inputs

a) An interface, either a Bridge Port, or an aggregated IEEE 802.3 port within a Bridge Port, on which MPs might be configured;
b) An MD Level;
c) A value indicating the direction in which any reported MP faces, either:
   1) Down (Active SAP is further away from the Frame filtering entity); or
   2) Up (Active SAP is closer to the Frame filtering entity); and
d) A specific VID to which the MEPs and MHFs are attached, or 0, for those attached to no VID.

12.14.2.1.3 Outputs

a) Operation status. This takes one of the following values:
   1) Operation rejected due to illegal inputs;
   2) No MP is configured on the indicated Bridge Port or aggregated port at the indicated MD Level and direction;
   3) A MEP is configured; or
   4) An MHF is configured;
b) Either the Maintenance Domain managed object (12.14.5) to which the MP's MA is associated, or an indication that there is no Maintenance Domain associated with the MP;
c) Either the Maintenance Association managed object (12.14.6) to which the MP is associated, or an indication that there is no MA associated with the MP;
d) If a MEP is configured, its MEPID, else 0; and
e) The MAC address of the MP.

12.14.3 Default MD Level managed object

There is a single Default MD Level managed object per Bridge. It controls MIP Half Function (MHF) creation for VIDs that are not contained in the list of VIDs attached to any specific Maintenance Association managed object [item c) in 12.14.5.3.2], and the transmission of the Sender ID TLV (21.5.3) by those MHFs.

The management commands that can be performed on the Default MD Level managed object are as follows:

a) Read Default MD Level managed object (12.14.3.1); and
b) Write Default MD Level managed object (12.14.3.2).

12.14.3.1 Read Default MD Level managed object

12.14.3.1.1 Purpose

To read the table of default parameters for controlling MHFs on VIDs not associated with any MA.

12.14.3.1.2 Input

a) A VID.

12.14.3.1.3 Output

a) A list of VIDs associated with any MHF on the VID named in item a) in 12.14.3.1.2, always including that VID. The first VID is the MAs' Primary VID. List is empty if no VID specified in 12.14.3.1.2;
b) A Boolean value indicating whether this entry is in effect or has been overridden by the existence of a Maintenance Association managed object associated with the same VID and MD Level, and on which is configured an Up MEP. True if this Maintenance Domain managed object is in effect;
c) (writable) The MD Level at which MHFs are to be created;
d) (writable) An enumerated value indicating whether the management entity can create MHFs for this VID(s), either defMHFnone (1, the default), defMHFdefault (2), or defMHFexplicit (3) (22.2.3); and
e) (writable) An enumerated value indicating what, if anything, is to be included in the Sender ID TLV (21.5.3) transmitted by MPs configured in the Default Maintenance Domain, either sendIdNone (1, the default value), sendIdChassis (2), sendIdManage (3), or sendIdChassisManage (4).

12.14.3.2 Write Default MD Level managed object

12.14.3.2.1 Purpose

To write entries in the table of default parameters for controlling MHFs on VIDs not associated with any MA.

12.14.3.2.2 Inputs

a) A list of VIDs, or 0, indicating no VID. The first VID in the list is the Primary VID;
b) One of the writable managed object in 12.14.3.1.3.

12.14.3.2.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected because creating or assigning these VIDs to this MD Level would exceed the number of MD Levels supported by some Bridge Port (item c) in 22.2.4);
    2) Operation rejected because the VID list provided [item a) in 12.14.3.2.2] specifies a different Primary VID, or contains one or more but not all of the VIDs, in the definition of an MA or some other entry in the Default MD Level managed object; or
    3) Operation accepted.

12.14.4 Configuration Error List managed object

The Configuration Error List managed object provides a list of Bridge Ports, aggregated ports, and VIDs that are incorrectly configured.

12.14.4.1 Read Configuration Error List managed object

12.14.4.1.1 Purpose

To read entries in the list of Bridge Ports or aggregated ports and VIDs currently configured in error. The Configuration Error List managed object lists only those Bridge Ports or aggregated ports and vlan_identifiers that have configuration errors.

12.14.4.1.2 Inputs

a) A vlan_identifier specifying which VLAN to check for Bridge Ports or aggregated ports in error; and
b) An interface, either a Bridge Port or an aggregated IEEE 802.3 port within a Bridge Port.

12.14.4.1.3 Outputs

a) An indication of the result of the operation, either:
    1) Operation rejected due to illegal vlan_identifier;
    2) Operation rejected due to illegal or non-existent Bridge Port or aggregated port; or
    3) Operation accepted; and
b) If the operation was accepted, a vector of Boolean error conditions from 22.2.4, any of which may be true:
    1) CFMleak;
    2) ConflictingVIDs;
    3) ExcessiveLevels; and/or
    4) OverlappedLevels.

12.14.5 Maintenance Domain managed object

There can be any number of Maintenance Domain managed objects per Bridge. A Maintenance Domain managed object is required in order to create an MA with a MAID that includes that Maintenance Domain's Name. From this Maintenance Domain managed object, all Maintenance Association managed objects associated with that Maintenance Domain managed object can be accessed, and thus controlled.

The management operations that can be performed on the Maintenance Domain managed object are as follows:

a) Read Maintenance Domain managed object (12.14.5.1);
b) Write Maintenance Domain managed object (12.14.5.2);
c) Create Maintenance Association managed object (12.14.5.3); and
d) Delete Maintenance Association managed object (12.14.5.4).

NOTE 1—The Maintenance Domain managed object is defined in such a manner that it can be configured identically in all of the Bridges in that Bridged Network. All of the information local to a particular Maintenance Association or a particular Bridge is contained in other managed objects, e.g., the Maintenance Association managed object (12.14.6) or the Maintenance association End Point managed object (12.14.7).

NOTE 2—A single Maintenance Domain managed object can be used to manage any number of separate Maintenance Domains, as Maintenance Domains are defined in 18.1. For example, the Maintenance Association managed object for all of the physical links in a Bridged Network that are protected by MAs can be grouped together under a single Maintenance Domain managed object for the convenience of the network administrator.

NOTE 3—Multiple Maintenance Domain managed objects can be used to manage a single Maintenance Domain, as Maintenance Domains are defined in 18.1. For example, a Provider Bridged Network's DoSAPs could be split among two Maintenance Domain managed objects, one for point-to-point services and one for multipoint services.

12.14.5.1 Read Maintenance Domain managed object

12.14.5.1.1 Purpose

To obtain information from a Bridge about a specific Maintenance Domain managed object.

12.14.5.1.2 Inputs

a) A reference to a particular Maintenance Domain managed object (12.14.5).

12.14.5.1.3 Outputs

a) Operation status. This takes one of the following values:
   1) Operation rejected due to nonexistent Maintenance Domain; or
   2) Operation accepted;
b) The MD Level of the specific Maintenance Domain identified by the Input;
c) (writable) An enumerated value indicating whether the management entity can create MHFs for this Maintenance Domain, either defMHFnone (1, the default value), defMHFdefault (2), or defMHFexplicit (3) (22.2.3);
d) (writable) An enumerated value indicating what, if anything, is to be included in the Sender ID TLV (21.5.3) transmitted by MPs configured in this Maintenance Domain, either sendIdNone (1, the default value), sendIdChassis (2), sendIdManage (3), or sendIdChassisManage (4) [item d) in 12.14.6.1.3];
e) (writable) The network address to which Fault Alarms (12.14.7.7) are to be transmitted, or a value indicating "Fault Alarms are not to be transmitted." Default value is "not transmitted"; and

NOTE—The variables c) through e) could be writable by the organization operating the Maintenance Domain.

f) A list of references to the Maintenance Association managed objects (12.14.6) attached to the indicated Maintenance Domain managed object.

12.14.5.2 Write Maintenance Domain managed object

12.14.5.2.1 Purpose

To alter a value of a specific Maintenance Domain managed object.

12.14.5.2.2 Inputs

a) A reference to a particular Maintenance Domain managed object (12.14.5);
b) A reference to a writable managed object in 12.14.5.1.3 to be altered; and
c) A new value for the managed object.

12.14.5.2.3 Outputs

a) Operation status. This takes one of the following values:
   1) Operation rejected due to nonexistent Maintenance Domain;
   2) Operation rejected due to the selected managed object being Read-Only;
   3) Operation rejected due to invalid value for the selected managed object; or
   4) Operation accepted.

12.14.5.3 Create Maintenance Association managed object

12.14.5.3.1 Purpose

To create a new Maintenance Association managed object within a Maintenance Domain managed object.

12.14.5.3.2 Inputs

a) A reference to a particular Maintenance Domain managed object (12.14.5);
b) The Short MA Name of an MA within that Maintenance Domain, including the format specifier from Table 21-20 (21.6.5.4), default value is a 2-octet integer (format 3) containing the primary VID, or 0, if the MA is not attached to a VID; and
c) The list of VIDs monitored by this MA, or 0, if the MA is not attached to a VID. The first VID in the list is the MA's Primary VID (default none).

12.14.5.3.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent Maintenance Domain;
    2) Operation rejected due to invalid Short MA Name (21.6.5.4);
    3) Operation rejected due to the existence of another MA in the same Maintenance Domain with the same Short MA Name;
    4) Operation rejected because creation of this MA would exceed the number of MD Levels supported by some Bridge Port [item c) in 22.2.4];
    5) Operation rejected because the VID list provided [item c) in 12.14.5.3.2] specifies a different Primary VID, or contains one or more, but not all, of the VIDs, in the definition of another MA or an entry in the Default MD Level managed object; or
    6) Operation accepted.

12.14.5.4 Delete Maintenance Association managed object

12.14.5.4.1 Purpose

To delete a specific Maintenance Association managed object from a Maintenance Domain managed object.

12.14.5.4.2 Inputs

a) A reference to a particular Maintenance Association managed object (12.14.6).

12.14.5.4.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent MA; or
    2) Operation accepted.

12.14.6 Maintenance Association managed object

There can be any number of Maintenance Association managed objects per Bridge, one for each service instance for which an MP is defined on that Bridge. From this Maintenance Association managed object, all Maintenance association End Point managed objects associated with that Maintenance Association managed object can be accessed, and thus controlled.

The management operations that can be performed on the Maintenance Association managed object are as follows:

a) Read Maintenance Association managed object (12.14.6.1);
b) Write Maintenance Association managed object (12.14.6.2);
c) Create Maintenance association End Point managed object (12.14.6.3); and
d) Delete Maintenance association End Point managed object (12.14.6.4).

NOTE 1—The Maintenance Association managed object is defined in such a manner that it can be configured identically in all of the Bridges in its Maintenance Domain. All information local to a particular Bridge is contained in other managed objects, e.g., the Maintenance association End Point managed object (12.14.7).

NOTE 2—Although MIP Half Functions (MHFs) are created via the Maintenance Domain managed object, the Default MD Level managed object, and the Maintenance Association managed object, and although there is a Maintenance association End Point managed object, there is no "MIP Half Function managed object." One reason for this omission is that no controls over the behavior of the MHF beyond its existence is required. Another is that the burden of implementing, providing access to, and providing the maintenance capabilities to access the counters and similar managed objects appropriate to an MHF Managed Object seemed to the developers of this standard to outweigh the benefits of having that information available.

12.14.6.1 Read Maintenance Association managed object

12.14.6.1.1 Purpose

To obtain information from a Bridge about a specific Maintenance Association managed object.

12.14.6.1.2 Inputs

a) A reference to a particular Maintenance Association managed object (12.14.6).

12.14.6.1.3 Outputs

a) Operation status. This takes one of the following values:
   1) Operation rejected due to nonexistent MA; or
   2) Operation accepted;
b) The VID(s) monitored by this MA, or 0, if the MA is not attached to any VID. The first VID returned is the MA's Primary VID;
c) (writable) An enumerated value indicating whether the management entity can create MHFs for this MA, either defMHFnone (1), defMHFdefault (2), defMHFexplicit (3), or defMHFdefer (4), (22.2.3), default value is defMHFdefer;
d) (writable) An enumerated value indicating what, if anything, is to be included in the Sender ID TLV (21.5.3) transmitted by MPs configured in this MA, either:
   1) **sendIdNone:** The Sender ID TLV is not to be sent;
   2) **sendIdChassis:** The Chassis ID Length, Chassis ID Subtype, and Chassis ID fields of the Sender ID TLV are to be sent, but not the Management Address Length or Management Address fields;
   3) **sendIdManage:** The Management Address Length and Management Address of the Sender ID TLV are to be sent, but the Chassis ID Length is to be transmitted with a 0 value, and the Chassis ID Subtype and Chassis ID fields not sent;
   4) **sendIdChassisManage:** The Chassis ID Length, Chassis ID Subtype, Chassis ID, Management Address Length, and Management Address fields are all to be sent; or
   5) **sendIdDefer:** (the default value) The contents of the Sender ID TLV are determined by the Maintenance Domain managed object, item d) in 12.14.5.1.3;
e) (writable) An enumerated value, other than 0, indicating the interval between CCM transmissions to be used by all MEPs in the MA (20.8.1, 21.6.1.3) (default 1 s);
f) (writable) The network address to which Fault Alarms (12.14.7.7) are to be transmitted, a value indicating "not specified," or a value indicating "Fault Alarms are not to be transmitted." If "not specified," the address used is that from the Maintenance Domain managed object [item e) in 12.14.5.1.3]; and

g) (writable) A list of the MEPIDs of the MEPs in the MA.

12.14.6.2 Write Maintenance Association managed object

12.14.6.2.1 Purpose

To alter a value of a specific Maintenance Association managed object.

12.14.6.2.2 Inputs

a) A reference to a particular Maintenance Association managed object (12.14.6);
b) A reference to one of the writable managed objects in 12.14.6.1.3 is to be altered; and
c) A new value for the managed object.

12.14.6.2.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent MA;
    2) Operation rejected due to the selected managed object being Read-Only;
    3) Operation rejected due to lack of authority to set this variable;
    4) Operation rejected due to invalid value for the selected managed object; or
    5) Operation accepted.

12.14.6.3 Create Maintenance association End Point managed object

12.14.6.3.1 Purpose

To create a new Maintenance association End Point managed object within a Maintenance Association managed object on a specific Bridge Port or aggregated port.

12.14.6.3.2 Inputs

a) A reference to a particular Maintenance Association managed object (12.14.6);
b) A MEPID for the created MEP (default 1);
c) A value indicating the direction in which the MEP faces on the Bridge Port or aggregated port, either:
    1) Down (Active SAP is further away from the Frame filtering entity) (the default); or
    2) Up (Active SAP is closer to the Frame filtering entity); and
d) An interface, either a Bridge Port or an aggregated IEEE 802.3 port within a Bridge Port.

12.14.6.3.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent MA;
    2) Operation rejected because the specified MEPID already exists in this MA in this Bridge;
    3) Operation rejected because MEPID is not in the MA's list of MEPIDs [item g) in 12.14.6.1.3];
    4) Operation rejected due to the existence of a MEP in the same direction (Up or Down) at that same MD Level, for the same VID(s) as this MEP (or for no VID, if this MEP's MA has no VID), on that Bridge Port or aggregated IEEE 802.3 port;
    5) Operation failed because one or more of the listed VIDs in this MA are assigned to some other MA on which Up MEPs are configured on any Bridge Port;
    6) Operation rejected because the Bridge is incapable of instantiating a MEP on that Bridge Port or aggregated IEEE 802.3 port;
    7) Operation rejected because an Up MEP cannot be configured for an MA that has no VID;

    8) Operation rejected because a MEP exists on this MA that has a different value for its Up/Down parameter [item c) in 12.14.6.3.2]; or
    9) Operation accepted.

12.14.6.4 Delete Maintenance association End Point managed object

12.14.6.4.1 Purpose

To delete an existing Maintenance association End Point managed object from a Maintenance Association managed object.

12.14.6.4.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7).

12.14.6.4.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent MEP; or
    2) Operation accepted.

12.14.7 Maintenance association End Point managed object

There can be any number of Maintenance association End Point managed objects per Bridge, one for each MEP defined within that Bridge. From this Maintenance association End Point managed object, all management objects related to that MEP can be controlled.

The management operations that can be performed on the Maintenance association End Point managed object are as follows:

a) Read Maintenance association End Point managed object (12.14.7.1);
b) Write Maintenance association End Point managed object (12.14.7.2);
c) Transmit Loopback Messages (12.14.7.3);
d) Transmit Linktrace Message (12.14.7.4);
e) Read Linktrace Reply (12.14.7.5);
f) Read MEP Database (12.14.7.6); and
g) Transmit MEP Fault Alarm (12.14.7.7).

12.14.7.1 Read Maintenance association End Point managed object

12.14.7.1.1 Purpose

To obtain information from a Bridge about a specific Maintenance association End Point managed object.

12.14.7.1.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7).

12.14.7.1.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent MEP; or
    2) Operation accepted;

b) An interface, either a Bridge Port or an aggregated IEEE 802.3 port within a Bridge Port, to which the MEP is attached;
c) A value indicating the direction in which the MEP faces on the interface, either:
   1) Down (Active SAP is further away from Frame filtering entity); or
   2) Up (Active SAP is closer to the Frame filtering entity);
d) (writable) An integer indicating the Primary VID of the MEP, always one of the VIDs assigned to the MEP's MA. The value 0 indicates that either the Primary VID is that of the MEP's MA or that the MEP's MA is associated with no VID (20.9.7);
e) (writable) A Boolean flag indicating the administrative state of the MEP (20.9.1) (default false);
f) A value indicating the current state of the MEP Fault Notification Generator state machine (20.35);
g) (writable) A Boolean flag indicating whether the MEP is or is not to generate CCMs (CCIenabled, 20.10.1) (default false, no CCMs);
h) (writable) The priority parameter for CCMs and LTMs transmitted by the MEP (default value: the highest priority, i.e., that with the highest numerical value, allowed to pass through the Bridge Port for any of this MEP's VIDs);

NOTE—The management entity can obtain the default value for this variable from the Priority regeneration table in 6.9.3 by extracting the highest priority value in the table on this MEP's Bridge Port (1 is lowest, then 2, then 0, then 3–7), appropriate to the direction in which frames are emitted from the MEP's Active SAP.

i) The MAC address of the MEP (19.4);
j) (writable) The network address to which Fault Alarms (12.14.7.7) are to be transmitted, a value indicating "not specified," or a value indicating "Fault Alarms are not to be transmitted." If "not specified," the address used is that from the Maintenance Association managed object [item f) in 12.14.6.1.3];
k) (writable) An integer value specifying the lowest priority defect that is allowed to generate a Fault Alarm (20.9.5), either:
   1) All defects: DefRDICCM, DefMACstatus**,** DefRemoteCCM, DefErrorCCM, and DefXconCCM;
   2) Only DefMACstatus**,** DefRemoteCCM, DefErrorCCM, and DefXconCCM (the default);
   3) Only DefRemoteCCM, DefErrorCCM, and DefXconCCM;
   4) Only DefErrorCCM and DefXconCCM;
   5) Only DefXconCCM; or
   6) No defects are to be reported;
l) (writable) The time that defects must be present before a Fault Alarm is issued (20.33.3) (default 2.5 s);
m) (writable) The time that defects must be absent before resetting a Fault Alarm (20.33.4) (default 10 s);
n) An enumerated value indicating the highest-priority defect that has been present since the MEP Fault Notification Generator state machine was last in the FNG_RESET state (20.33.9), either:
   1) **DefNone**: No defect has been present since the last FNG_RESET state;
   2) **DefRDICCM**: The last CCM received by this MEP from some remote MEP contained the RDI bit, so variable item o) in 12.14.7.1.3 is set;
   3) **DefMACstatus**: The last CCM received by this MEP from some remote MEP indicated that the transmitting MEP's associated MAC is reporting an error status via the Port Status TLV (21.5.4) or Interface Status TLV (21.5.5), so variable item p) in 12.14.7.1.3 is set;
   4) **DefRemoteCCM**: This MEP is not receiving CCMs from some other MEP in its configured list [item g) in 12.14.6.1.3], so variable item q) in 12.14.7.1.3 is set;
   5) **DefErrorCCM**: This MEP is receiving invalid CCMs, so variable item r) in 12.14.7.1.3, is set; or
   6) **DefXconCCM**: This MEP is receiving CCMs that could be from some other MA, so variable item s) in 12.14.7.1.3 is set;
o) A Boolean flag (20.33.7) indicating that some other MEP in this MEP's MA is transmitting the RDI bit (can trigger DefRDICCM);

p) A Boolean flag (20.33.6) indicating that a Port Status TLV (21.5.4) or Interface Status TLV (21.5.5) is indicating an error condition (can trigger DefMACstatus);
q) A Boolean flag (20.33.5) indicating that CCMs are not being received from at least one of the configured remote MEPs (can trigger DefRemoteCCM);
r) A Boolean flag (20.21.3) indicating that erroneous CCMs are being received from some MEP in this MEP's MA (can trigger DefErrorCCM);
s) A Boolean flag (20.23.3) indicating that CCMs are being received from a MEP that could be in some other MA (can trigger DefXconCCM);
t) The last-received CCM (20.21.2) that triggered a DefErrorCCM fault (20.21.3);
u) The last-received CCM (20.23.2) that triggered a DefXconCCM fault (20.23.3);
v) (optional) The total number of out-of-sequence CCMs received (20.16.12);
w) The total number of CCMs transmitted (20.10.2);
x) The next Loopback Transaction Identifier to be sent in an LBM (20.28.2);
y) The total number of valid, in-order LBRs received (20.31.1);
z) The total number of valid, out-of-order LBRs received (20.31.1);
aa) (optional) The total number of LBRs received whose mac_service_data_unit did not match (except for the OpCode) that of the corresponding LBM (20.2.3);
ab) The next LTM Transaction Identifier to be sent in an LTM (20.36.1);
ac) The total number of unexpected LTRs received (20.39.1); and
ad) The total number of LBRs transmitted (20.26.2).

12.14.7.2 Write Maintenance association End Point managed object

12.14.7.2.1 Purpose

To alter a value in a specific managed object within a specific Maintenance association End Point managed object.

12.14.7.2.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7);
b) A reference to one of the writable managed objects in 12.14.7.1.3 is to be altered; and
c) A new value for the managed object.

12.14.7.2.3 Outputs

a) Operation status. This takes one of the following values:
    1) Operation rejected due to nonexistent MEP;
    2) Operation rejected due to the selected managed object being Read-Only;
    3) Operation rejected due to invalid value for the selected managed object; or
    4) Operation accepted.

12.14.7.3 Transmit Loopback Messages

12.14.7.3.1 Purpose

To signal to the MEP to transmit some number of LBMs.

NOTE—The Maintenance association End Point managed object (12.14.7) can be examined to determine whether or not the corresponding LBRs have been received by the MEP.

12.14.7.3.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7);

b) An indication of the destination MAC address for LBMs transmitted by the MEP (20.29.1), either:
   1) The MEPID of a MEP in that MA; or
   2) An Individual destination MAC address;
c) The number of LBMs to be transmitted (default 1);
d) An arbitrary amount of data to be included in a Data TLV, along with an indication whether the Data TLV is to be included (20.29.1) (default no Data TLV); and
e) The priority and drop_eligible parameters to be used in the transmitted LBMs [default is CCM priority item h) in 12.14.7.1.3, drop_eligible false].

12.14.7.3.3 Outputs

a) An indication of whether the command was accepted by the MEP, either:
   1) Operation rejected due to nonexistent MEP;
   2) Operation rejected due to invalid number of LBMs to be transmitted;
   3) Operation rejected due to too much data to be transmitted;
   4) Operation rejected due to invalid priority or drop_eligible parameters to be transmitted;
   5) The LBM(s) will be (or have been) sent; or
   6) The LBM(s) will not be sent (e.g., because the Bridge is busy with another LBM transmit operation); and
b) The Loopback Transaction Identifier (20.28.2) of the first LBM (to be) sent. The value returned is undefined if the Transmit Loopback Messages command was not accepted.

12.14.7.4 Transmit Linktrace Message

12.14.7.4.1 Purpose

To signal to the MEP to transmit an LTM and to create an LTM entry in the MEP's Linktrace Database.

NOTE—The Maintenance association End Point managed object (12.14.7) can be examined to determine whether or not the corresponding LTRs have been received by the MEP.

12.14.7.4.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7);
b) The Flags field for LTMs transmitted by the MEP (20.37.1);
c) An indication of the Target MAC Address field to be transmitted, either:
   1) The MEPID of another MEP in the same MA; or
   2) An Individual destination MAC address; and
d) An initial value for the LTM TTL field (21.8.4). Default value, if not specified, is 64.

12.14.7.4.3 Outputs

a) An indication of whether the command was accepted by the MEP, either:
   1) Operation rejected due to nonexistent MEP;
   2) Operation rejected due to invalid Flags field to be transmitted;
   3) Operation rejected due to invalid Target MAC Address to be transmitted;
   4) Operation rejected due to invalid LTM TTL field to be transmitted;
   5) The command was accepted and the LTM has (or will be) transmitted; or
   6) The command was rejected and no LTM will be sent (e.g., because the Bridge is busy with another LTM transmit operation);
b) The LTM Transaction Identifier (20.36.1) of the LTM sent. The value returned is undefined if the Transmit Linktrace Message command was not accepted; and
c) The LTM Egress Identifier TLV value transmitted in the LTM. The value returned is undefined if the Transmit Linktrace Message command was not accepted.

12.14.7.5 Read Linktrace Reply

12.14.7.5.1 Purpose

To obtain from the MEP's Linktrace database (20.36.2) the LTM entry for a previously transmitted LTM with a specific LTM Transaction Identifier. An LTM entry consists of a list of LTR entries, each corresponding to a Linktrace Reply (LTR) PDU received in response to that LTM. The LTM Transaction Identifier returned by successful Transmit Linktrace Message command [item b) in 12.14.7.4.3] is used in the Read Linktrace Reply command [item b) in 12.14.7.5.2] to select the LTM entry.

See J.5 for a discussion of how to interpret the output from the Read Linktrace Reply command.

12.14.7.5.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7);
b) The LTM Transaction Identifier returned from a previous Transmit Linktrace Message command, indicating the LTM entry to which the LTR entries will be attached; and
c) An index to distinguish among multiple LTRs with the same LTR Transaction Identifier field value.

12.14.7.5.3 Outputs

A list of LTR entries returned corresponding to the selected LTM Transaction Identifier (20.36.2). Each LTR entry in the list returns:

a) An indication of whether the command was accepted by the MEP, either:
    1) Operation rejected due to nonexistent MEP;
    2) Operation rejected due to LTM Transaction Identifier not found;
    3) Operation rejected due to LTR entry has been discarded because too many LTRs have been returned corresponding to that LTM Transaction Identifier;
    4) Operation rejected due to LTR entry with that index has not yet been returned; or
    5) Operation accepted;
b) The integer Reply TTL field value returned in the LTR (20.36.2.2);
c) A Boolean value stating whether an LTM was forwarded by the responding MP, as returned in the FwdYes flag of the Flags field (20.36.2.1);
d) A Boolean value stating whether the forwarded LTM reached a MEP enclosing its MA, as returned in the TerminalMEP flag of the Flags field (20.36.2.1);
e) An octet string holding the Last Egress Identifier field returned in the LTR Egress Identifier TLV of the LTR (20.36.2.3);
f) An octet string holding the Next Egress Identifier field returned in the LTR Egress Identifier TLV of the LTR (20.36.2.4);
g) An enumerated value indicating the value returned in the Relay Action field (20.36.2.5) either RlyHit, RlyFDB, or RlyMPDB;
h) (if returned in the LTR) An enumerated value indicating the format of the Chassis ID (21.5.3.2);
i) (if returned in the LTR) The Chassis ID (21.5.3.3);
j) (if returned in the LTR) The Management Address information of the Bridge transmitting the LTR (21.5.3.7);
k) (if returned in the LTR) An enumerated value indicating the value returned in the Ingress Action field (20.36.2.6) of the LTR, either IngOK, IngDown, IngBlocked, or IngVID;
l) (if returned in the LTR) The MAC address returned in the Ingress MAC Address field (20.36.2.7);
m) (if returned in the LTR) An enumerated value indicating the format of the Ingress Port ID field (20.36.2.8);
n) (if returned in the LTR) The Ingress Port ID (20.36.2.9);
o) (if returned in the LTR) An enumerated value indicating the value returned in the Egress Action field (20.36.2.10) of the LTR, either EgrOK, EgrDown, EgrBlocked, or EgrVID;

p) (if returned in the LTR) The MAC address returned in the Egress MAC Address field (20.36.2.11);
q) (if returned in the LTR) An enumerated value indicating the format of the Egress Port ID (20.36.2.12);
r) (if returned in the LTR) The Egress Port ID (20.36.2.13); and
s) (if returned in the LTR) The OUI and contents of any Organization-Specific TLVs (21.5.2) returned in the LTR.

12.14.7.6 Read MEP Database

12.14.7.6.1 Purpose

To obtain from the MEP the contents of one element in the MEP CCM Database.

12.14.7.6.2 Inputs

a) A reference to a particular Maintenance association End Point managed object (12.14.7).;
b) The MEPID of a remote MEP whose information from the selected database is to be returned.

12.14.7.6.3 Outputs

a) An indication of whether the command was accepted by the MEP, either:
   1) Operation rejected due to nonexistent MEP;
   2) Operation rejected due remote MEPID not configured in MA; or
   3) Operation accepted;
b) An enumerated value indicating the operational state of the Remote MEP state machine (20.20) for this remote MEP either:
   1) RMEP_IDLE;
   2) RMEP_START;
   3) RMEP_FAILED; or
   4) RMEP_OK;
c) The time (SysUpTime, IETF RFC 3418) at which the Remote MEP state machine last entered either the RMEP_FAILED or RMEP_OK state, or 0 if it has not yet entered either of those states;
d) The MAC address of the remote MEP (the source_address of the last received CCM) or 0 if no CCM has been received (20.19.7);
e) A Boolean value indicating the state of the RDI bit in the last received CCM (true for RDI = 1), or false, if none has been received (20.19.2);
f) (optional) The enumerated value from the Port Status TLV (20.19.3) from the last CCM received from the remote MEP, or the default value:
   0) **psNoPortStateTLV:** Indicates either that no CCM has been received, or that no Port Status TLV was present in the last CCM received;
g) (optional) The enumerated value from the Interface Status TLV (20.19.4) from the last CCM received from the remote MEP, or the default value:
   0) **isNoInterfaceStatusTLV:** Indicates either that no CCM has been received or that no Interface Status TLV was present in the last CCM received; and
h) (optional) The Sender ID TLV (20.19.5) from the last CCM received from the remote MEP or the default value:
   0) **isNoSenderIdTLV:** Indicates either that no CCM has been received, or that no Sender ID TLV was present in the last CCM received.

12.14.7.7 Transmit MEP Fault Alarm

12.14.7.7.1 Purpose

To alert the Manager to the existence of a fault in a monitored MA by issuing a Fault Alarm. Transmit MEP Fault Alarm is not a command; it is an unsolicited notification from the xmitFaultAlarm() procedure, issued upon detection of the Fault Alarm condition by the MEP Fault Notification Generator state machine. The Fault Alarm is sent to the address specified in item j) in 12.14.7.1.3.

12.14.7.7.2 Outputs

a) An indication of which Bridge is reporting the Fault Alarm;
b) A reference, in a suitable format for the method of delivering the Fault Alarm, to the reporting MEP; and
c) An enumerated value, equal to the contents of the variable highestDefect (20.33.9), and to the corresponding MEP managed object [item n) in 12.14.7.1.3].

15. Support of the MAC Service by Provider Bridged Networks

Insert the following at the end of Clause 15:

15.10 Connectivity Fault Management

The most common use of bridged networks as described in IEEE Std 802.1D, and this standard IEEE Std 802.1Q, has been in an environment where a single administration operates the network and where physical access to all bridges is available. A service instance, on the other hand, can be provided to a customer by more than one interconnected Provider Bridged Network. Furthermore, each network can be under different and independent administrative control, each with restricted management access to each other's equipment. Physical access to Provider Bridges and other network equipment can be difficult and expensive; equipment can be many kilometers from the service personnel and can be positioned underground, atop a tower, or on the customer's premises. As a result, the methods commonly used to ensure the availability of the services offered by IEEE 802.1D and IEEE 802.1Q bridged networks, which assume a single administration and easy access to equipment, are inadequate to manage interconnected Provider Bridged Networks.

CFM defines a means to address these difficulties, providing a means whereby diverse administrations can detect, isolate, and correct connectivity faults in the MAC Service with a minimum of access to each others' equipment. CFM is defined in five clauses:

- Clause 18 introduces the principles of CFM.
- Clause 19 defines the entities comprising CFM.
- Clause 20 defines the protocols exchanged by those entities.
- Clause 21 defines the formats of the various CFM PDUs.
- Clause 22 illustrates the usage of CFM in actual networks.

17. Management protocol

Delete the existing text of Clause 17.

~~NOTE—At the time of publication of this standard, MIB modules for management of various aspects of VLAN Bridge operation, including object definitions for MRP, MMRP, and MVRP were under development as part of project P802.1ap.~~

Insert the following new Clause 17.

NOTE—At the time of publication of this standard, MIB modules for management of various aspects of VLAN Bridge operation were under development as part of projects IEEE P802.1ah and IEEE P802.1ap. These projects will result in additional material being incorporated into this clause. At present, the clause deals only with management of CFM.

17.1 Internet Standard Management Framework

For a detailed overview of the documents that describe the current Internet Standard Management Framework, please refer to section 7 of IETF RFC 3410 (2002) [B7].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This clause specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, IETF RFC 2578, IETF RFC 2579, and IETF RFC 2580.

Control of the transmission of Fault Alarms, which are Notifications in SNMP, are described by STD 62, IETF RFC 3413, IETF RFC 3417, and IETF RFC 4789. As a consequence of the concentration of control of the reporting of SNMP Notifications in the SNMP-TARGET-MIB and SNMP-NOTIFICATION-MIB of RFC 3413, the variables controlling Fault Alarm notifications described in Clause 12 [item e) in 12.14.5.1.3, item f) in 12.14.6.1.3, and item j) in 12.14.7.1.3] require no specific objects in the CFM MIB module in 17.5.

17.2 Relationship with the managed objects definitions

Subclause 12.14 of this document defines the information model associated with this standard in a protocol independent manner. Table 17-1 describes the relationship between the SMIv2 objects defined in the MIB module in 17.5, the variables defined in Clause 20, and the protocol-independent objects defined in 12.14.

Table 17-1—Correspondence between variables, managed objects, and MIB objects

Variable (Clause 20)	Managed object (12.14)	MIB object (17.5)
CFM Stack managed object (12.14.2)		`dot1agCfmStackTable`
	12.14.2.1.2:a	`dot1agCfmStackifIndex`
	12.14.2.1.2:d	`dot1agCfmStackVlanIdOrNone`
	12.14.2.1.2:b	`dot1agCfmStackMdLevel`
	12.14.2.1.2:c	`dot1agCfmStackDirection`
	12.14.2.1.3:b	`dot1agCfmStackMdIndex`

Table 17-1—Correspondence between variables, managed objects, and MIB objects *(continued)*

Variable (Clause 20)	Managed object (12.14)	MIB object (17.5)
	12.14.2.1.2:c	`dot1agCfmStackMaIndex`
	12.14.2.1.3:d	`dot1agCfmStackMepId`
	12.14.2.1.3:e	`dot1agCfmStackMacAddress`
Default MD Level managed object (12.14.3)		`dot1agCfmDefaultMdTable`
	12.14.3.1.3:a, 12.14.3.2.2:a	`dot1agCfmDefaultMdPrimaryVid`
	12.14.3.1.3:b	`dot1agCfmDefaultMdStatus`
	12.14.3.1.3:c, 12.14.3.2.2:b	`dot1agCfmDefaultMdDefLevel`, `dot1agCfmDefaultMdLevel`
	12.14.3.1.3:d	`dot1agCfmDefaultMdDefMhfCreation`, `dot1agCfmDefaultMdMhfCreation`
	12.14.3.1.3:e	`dot1agCfmDefaultMdDefIdPermission`, `dot1agCfmDefaultMdIdPermission`
Configuration Error List managed object (12.14.4)		`dot1agCfmConfigErrorListTable`
	12.14.4.1.2:a	`dot1agCfmConfigErrorListVid`
	12.14.4.1.2:b	`dot1agCfmConfigErrorListIfIndex`
	12.14.4.1.3:b	`dot1agCfmConfigErrorListErrorType`
Maintenance Domain managed object (12.14.5)		`dot1agCfmMdTable`
	12.14.1.1.3:a:2	`dot1agCfmMdIndex`
	12.14.1.2.2:a	`dot1agCfmMdFormat, dot1agCfmMdName`
mdLevel (20.7.1)	12.14.5.1.3:b	`dot1agCfmMdMdLevel`
	12.14.5.1.3:c	`dot1agCfmMdMhfCreation`
	12.14.5.1.3:d	`dot1agCfmMdMhfIdPermission`
	12.14.5.1.3:e	`controlled by IETF RFC 3413`
	12.14.5.1.3:f	`dot1agCfmMdIndex`
Maintenance Association managed object (12.14.6)		`dot1agCfmMaNetTable`, `dot1agCfmMaCompTable` `dot1agCfmVlanTable`, `dot1agCfmMaMepListTable`
	12.14.6.1.2:a	`dot1agCfmMaIndex`
	12.14.5.3.2:b	`dot1agCfmMaNetFormat`, `dot1agCfmMaNetName`
	12.14.6.1.3:b	`dot1agCfmVlanVid`, `dot1agCfmMaCompNumberOfVids`
	12.14.6.1.3:c	`dot1agCfmMaCompMhfCreation`

Table 17-1—Correspondence between variables, managed objects, and MIB objects *(continued)*

Variable (Clause 20)	Managed object (12.14)	MIB object (17.5)
	12.14.6.1.3:d	dot1agCfmMaCompIdPermission
CCMinterval (20.8.1)	12.14.6.1.3:e	dot1agCfmMaNetCcmInterval
	12.14.6.1.3:f	controlled by IETF RFC 3413
	12.14.6.1.3:g	dot1agCfmMaMepListIdentifier
MEP managed object (12.14.7)		dot1agCfmMepTable
	12.14.7.1.2:a	dot1agCfmMdIndex, dot1agCfmMaIndex, dot1agCfmMepIdentifier
	12.14.7.1.3:b	dot1agCfmMepIfIndex
	12.14.7.1.3:c	dot1agCfmMepDirection
	12.14.7.1.3:d	dot1agCfmMepPrimaryVid
MEPactive (20.9.1)	12.14.7.1.3:e	dot1agCfmMepActive
20.35	12.14.7.1.3:f	dot1agCfmMepFngState
CCIenabled (20.10.1)	12.14.7.1.3:g	dot1agCfmMepCciEnabled
	12.14.7.1.3:h	dot1agCfmMepCcmLtmPriority
	12.14.7.1.3:i	dot1agCfmMepMacAddress
	12.14.7.1.3:j	controlled by IETF RFC 3413
lowestAlarmPri (20.9.5)	12.14.7.1.3:k	dot1agCfmMepLowPrDef
fngAlarmTime (20.33.3)	12.14.7.1.3:l	dot1agCfmMepFngAlarmTime
fngResetTime (20.33.4)	12.14.7.1.3:m	dot1agCfmMepFngResetTime
highestDefect (20.33.9)	12.14.7.1.3:n	dot1agCfmMepHighestPrDefect
someRDIdefect (20.33.7)	12.14.7.1.3:o	dot1agCfmMepDefects
someMACstatusDefect (20.33.6)	12.14.7.1.3:p	
someRMEPCCMdefect (20.33.5)	12.14.7.1.3:q	
errorCCMdefect (20.21.3)	12.14.7.1.3:r	
xconCCMdefect (20.23.3)	12.14.7.1.3:s	
errorCCMlastFailure (20.21.2)	12.14.7.1.3:t	dot1agCfmMepErrorCcmLastFailure
xconCCMlastFailure (20.23.2)	12.14.7.1.3:u	dot1agCfmMepXconCcmLastFailure
CCMsequenceErrors (20.16.12)	12.14.7.1.3:v	dot1agCfmMepCcmSequenceErrors
CCIsentCCMs (20.10.2)	12.14.7.1.3:w	dot1agCfmMepCciSentCcms
nextLBMtransID (20.28.2)	12.14.7.1.3:x	dot1agCfmMepNextLbmTransId
	12.14.7.1.3:y	dot1agCfmMepLbrIn
	12.14.7.1.3:z	dot1agCfmMepLbrInOutOfOrder

Table 17-1—Correspondence between variables, managed objects, and MIB objects *(continued)*

Variable (Clause 20)	Managed object (12.14)	MIB object (17.5)
	12.14.7.1.3:aa	`dot1agCfmMepLbrBadMsdu`
nextLTMtransID (20.36.1)	12.14.7.1.3:ab	`dot1agCfmMepLtmNextSeqNumber`
	12.14.7.1.3:ac	`dot1agCfmMepUnexpLtrIn`
	12.14.7.1.3:ad	`dot1agCfmMepLbrOut`
Transmit Loopback Messages (12.14.7.3)		`dot1agCfmMepTable,` `dot1agCfmMepTransmitLbmStatus`
	12.14.7.3.2:a	`dot1agCfmMdIndex, dot1agCfmMaIndex,` `dot1agCfmMepIdentifier`
	12.14.7.3.2:b	`dot1agCfmMepTransmitLbmDestMacAddress,` `dot1agCfmMepTransmitLbmDestMepId,` `dot1agCfmMepTransmitLbmDestIsMepId`
	12.14.7.3.2:c	`dot1agCfmMepTransmitLbmMessages`
	12.14.7.3.2:d	`dot1agCfmMepTransmitLbmDataTlv`
	12.14.7.3.2:e	`dot1agCfmMepTransmitLbmVlanPriority,` `dot1agCfmMepTransmitLbmVlanDropEnable`
	12.14.7.3.3:a	`dot1agCfmMepTransmitLbmResultOK`
	12.14.7.3.3:b	`dot1agCfmMepTransmitLbmSeqNumber`
Transmit Linktrace Message (12.14.7.4)		`dot1agCfmMepTable`
	12.14.7.4.2:a	`dot1agCfmMdIndex, dot1agCfmMaIndex,` `dot1agCfmMepIdentifier`
	12.14.7.4.2:b	`dot1agCfmMepTransmitLtmFlags`
	12.14.7.4.2:c	`dot1agCfmMepTransmitLtmTargetMacAddress` `dot1agCfmMepTransmitLtmTargetMepId,` `dot1agCfmMepTransmitLtmTargetIsMepId`
	12.14.7.4.2:d	`dot1agCfmMepTransmitLtmTtl`
	12.14.7.4.3:a	`dot1agCfmMepTransmitLtmResult`
	12.14.7.4.3:b	`dot1agCfmMepTransmitLtmSeqNumber`
	12.14.7.4.3:c	`dot1agCfmMepTransmitLtmEgressIdentifier`
Read Linktrace Reply (12.14.7.5)		`dot1agCfmLtrTable`
	12.14.7.5.2:a	`dot1agCfmMdIndex, dot1agCfmMaIndex,` `dot1agCfmMepIdentifier`
	12.14.7.5.2:b	`dot1agCfmLtrSeqNumber`
	12.14.7.5.2:c	`dot1agCfmLtrReceiveOrder`
ltrReplyTTL (20.36.2.2)	12.14.7.5.3:b	`dot1agCfmLtrTtl`

Table 17-1—Correspondence between variables, managed objects, and MIB objects *(continued)*

Variable (Clause 20)	Managed object (12.14)	MIB object (17.5)
ltrFlags (20.36.2.1)	12.14.7.5.3:c	`dot1agCfmLtrForwarded`
	12.14.7.5.3:d	`dot1agCfmLtrTerminalMep`
ltrLastEgressId (20.36.2.3)	12.14.7.5.3:e	`dot1agCfmLtrLastEgressIdentifier`
ltrNextEgressId (20.36.2.4)	12.14.7.5.3:f	`dot1agCfmLtrNextEgressIdentifier`
ltrRelayAction (20.36.2.5)	12.14.7.5.3:g	`dot1agCfmLtrRelay`
ltrIngressAction (20.36.2.6)	12.14.7.5.3:k	`dot1agCfmLtrIngress`
ltrIngressAddress (20.36.2.7)	12.14.7.5.3:l	`dot1agCfmLtrIngressMac`
ltrIngressPortIdSubtype (20.36.2.8)	12.14.7.5.3:m	`dot1agCfmLtrIngressPortIdSubtype`
ltrIngressPortId (20.36.2.9)	12.14.7.5.3:n	`dot1agCfmLtrIngressPortId`
ltrEgressAction (20.36.2.10)	12.14.7.5.3:o	`dot1agCfmLtrEgress`
ltrEgressAddress (20.36.2.11)	12.14.7.5.3:p	`dot1agCfmLtrEgressMac`
ltrEgressPortIdSubtype (20.36.2.12)	12.14.7.5.3:q	`dot1agCfmLtrEgressPortIdSubtype`
ltrEgressPortId (20.36.2.13)	12.14.7.5.3:r	`dot1agCfmLtrEgressPortId`
ltrOrgSpecTlv (20.36.2.15)	12.14.7.5.3:s	`dot1agCfmLtrOrganizationSpecificTlv`
ltrSenderIdTlv (20.36.2.14)	12.14.7.5.3:h	`dot1agCfmLtrChassisIdSubtype`
	12.14.7.5.3:i	`dot1agCfmLtrChassisId`
	12.14.7.5.3:j	`dot1agCfmLtrManAddressDomain`, `dot1agCfmLtrManAddress`
Read MEP Database (12.14.7.6)		`dot1agCfmMepDbTable`
	12.14.7.6.2:a	`dot1agCfmMdIndex`, `dot1agCfmMaIndex`, `dot1agCfmMepIdentifier`
	12.14.7.6.2:b	`dot1agCfmMepDbRMepIdentifier`
20.20	12.14.7.6.3:b	`dot1agCfmMepDbRMepState`
	12.14.7.6.3:c	`dot1agCfmMepDbRMepFailedOkTime`
rMEPmacAddress (20.19.7)	12.14.7.6.3:d	`dot1agCfmMepDbMacAddress`
rMEPlastRDI (20.19.2)	12.14.7.6.3:e	`dot1agCfmMepDbRdi`
rMEPlastPortState (20.19.3)	12.14.7.6.3:f	`dot1agCfmMepDbPortStatusTlv`
rMEPlastInterfaceStatus (20.19.4)	12.14.7.6.3:g	`dot1agCfmMepDbInterfaceStatusTlv`
rMEPlastSenderId (20.19.5)	12.14.7.6.3:h	`dot1agCfmMepDbChassisIdSubtype`
		`dot1agCfmMepDbChassisId`
		`dot1agCfmMepDbManAddressDomain`, `dot1agCfmMepDbManAddress`

Table 17-1—Correspondence between variables, managed objects, and MIB objects *(continued)*

Variable (Clause 20)	Managed object (12.14)	MIB object (17.5)
Transmit MEP Fault Alarm (12.14.7.7)		`dot1agCfmFaultAlarm`
	12.14.7.7.2:b, 12.14.7.7.2:c	`dot1agCfmMepHighestPrDefect`

17.3 Relationship to other MIB modules

The MIB modules used to manage Bridges are defined in IETF RFC 4188, IETF RFC 4318, and IETF RFC 4363. A Bridge may support Clause 12 using these MIB modules.

Subclause 17.5 defines a CFM MIB module that supports 12.14. A system implementing the CFM MIB module in 17.5 shall also implement at least the System Group of the SNMPv2-MIB defined in IETF RFC 3418 and the Interfaces Group (the Interfaces MIB module, or IF-MIB) defined in IETF RFC 2863. The Interfaces Group has one conceptual row in a table for every interface in a system. Section 3.3 of IETF RFC 2863, the Interface MIB Evolution, defines hierarchical relationships among interfaces.

IETF RFC 2863 also requires that any MIB module that is an adjunct of the Interface Group clarify specific areas within the Interface MIB module. These areas were intentionally left vague in IETF RFC 2863 to avoid over constraining the MIB, thereby precluding management of certain media types. These areas are clarified in IETF RFC 4188, IETF RFC 4318, and IETF RFC 4363, which define the Bridge MIB modules. Even if a Bridge supports none of RFCs 4188, 4318, or 4363, if it supports the CFM MIB module, and hence, the Interfaces Group, the clarifications in RFCs 4188, 4318, and 4363 shall be applied to the Interfaces Group. The relationship between RFC 2863 and RFC 3418 interfaces and Bridge Ports is also described in IETF RFC 4188 and IETF RFC 4363. In addition, if both the CFM MIB module (17.5) and IEEE Std 802.3, Clause 43, are supported, a Bridge shall:

a) Assign each IEEE 802.3 port in the aggregation its own conceptual row in the IETF RFC 2863 IF-MIB; and
b) Assign one conceptual row in the IETF RFC 2863 IF-MIB that references the aggregated port. This may be the same as the interface identifying the Bridge Port. It shall not be the same as any of the IEEE 802.3 ports being aggregated.

Further MIB module relationships related to security are explained in 17.4.

17.4 Security

It is expected that, in some provider networks, management access to a Bridge can be granted, by an organization that owns and is responsible for the normal operations of the Bridge, to another organization that needs to configure and manage CFM entities in that Bridge. For example, a provider could sell a service to a customer that connects sites in cities thousands of kilometers apart. That provider might not have a physical presence in all of those cities (or any of them, for that matter) and therefore would need to contract with other providers to supply a number of interconnected Provider Bridged Networks that together are able to offer the service to the customer. The relationship between the provider and Owner is not necessarily cordial; there can be personal or financial incentives for misbehavior.

To support this scenario, the MIB module in 17.5 defines the Maintenance Domain table, dot1agCfmMdTable. Each row (dot1agCfmMdEntry) in this table corresponds to one Maintenance Domain managed object (12.14.5). A Maintenance Domain managed object can be created, read, and written by the Bridge's Owner. More limited access to that object can be granted to another provider, who thereby becomes that Maintenance Domain's administrator. Associated with each row in the Maintenance Domain table are any number of rows (dot1agCfmMaNetEntry and dot1agCfmMaCompEntry) in the Maintenance Association table, dot1agCfmMaNetTable and dot1agCfmMaCompTable. Each of the paired rows in these two tables corresponds to a Maintenance Association managed object (12.14.6). A Maintenance Domain administrator has the ability to create and alter these rows. Additional MIB tables are defined in 17.5 that correspond to the Maintenance Domain list managed object (12.14.1), the CFM Stack managed object (12.14.2), and the Default MD Level managed object (12.14.3), all of which can be used by an Owner of the Bridge, but not by a Maintenance Domain administrator.

The SNMPv3 framework (IETF STD 62) defines a securityName, which provides for secure identification of a user (or group of users) via an SNMPv3 Security Model. For ease of reference in this present standard, we define the "Owner" of a system (in particular, a Bridge) as a user whose "MIB view" includes READ-WRITE access to the System Group of the SNMPv2-MIB (IETF RFC 3418). A Management Domain administrator is then a user whose MIB view includes more limited access to the Maintenance Domain managed object, and full READ-WRITE access to its subordinate Maintenance Association managed objects (12.14.6), each of which is a pair of rows (dot1agCfmMaNetEntry and dot1agCfmMaCompEntry) in the two tables, dot1agCfmMaNetTable and dot1agCfmMaCompTable.

A Bridge can support IETF STD 62 (SNMPv3) in order to provide a secure means for confining access by Management Domain administrators to their corresponding Maintenance Association managed objects, and to prevent inappropriate access to the rest of the Bridge's management objects. If the SNMPv3 framework is not implemented in a Bridge, an Owner is defined, imprecisely, as an administrator of the Bridge, and a Management Domain administrator (even less precisely) as an administrator of one Management Domain.

Therefore, it is recommended that the implementors consider the security features as provided by the SNMPv3 framework, including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy). Specifically, a Bridge can use the User-based Security Model STD 62, IETF RFC 3414, and/or the View-based Access Control Model STD 62, IETF RFC 3415, for controlling access to the CFM managed objects. It is then a customer/user responsibility to ensure that the SNMP agent giving access to an instance of this MIB is properly configured to give access to the objects only to those principals (users) that have legitimate rights to GET or SET (change/create/delete) them. SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in the CFM MIB module.

The control of access by providers to other MIB modules, for example, the Interface Group, is not specified by this standard.

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. Table 17-2 lists the MIB tables and objects and their sensitivity/vulnerability.

There is no detailed list in this standard of all vulnerable objects with a MAX-ACCESS clause of read-write or read-create and of the security threats associated to their intentional or unintentional manipulation by write actions. The reason is that this standard takes the approach that objects within a maintenance domain are protected as if in a "walled garden," accessible only by administrators authorized for the respective Maintenance Domains. See Figure 12-1 for an illustration of this concept.

The only exceptions to this approach are the read-create MIB objects that define the maintenance domains themselves and their capabilities. The vulnerabilities associated with these objects are shown in Table 17-3.

Table 17-2—Sensitive managed objects: tables and notifications

Table or object	Reason for sensitivity to security considerations
`dot1agCfmDefaultMdTable`, `dot1agCfmVlanTable`	The integrity of the spanning tree(s) running in a network is dependent on certain configuration parameters, especially Static VLAN Registration Entries (8.8.2). The assignment of VIDs to particular service instances, and hence to Maintenance Associations, is controlled by the administrator responsible for the integrity of the spanning tree, namely the Owner of the Bridge.
`dot1agCfmMdTable`, `dot1agCfmMaNetTable`, `dot1agCfmMaCompTable`, `dot1agCfmVlanTable`, `dot1agCfmMaMepListTable`, `dot1agCfmMepTable`, `dot1agCfmMepDbTable`, `dot1agCfmLtrTable`, `dot1agCfmFaultAlarm`	The business relationships among the various Maintenance Domain administrators, and between them and the Owner of the bridge, are not defined by this standard. The ability of one Maintenance Domain administrator to identify another, or even detect the presence of another Maintenance Domain administrator, in the same Bridge, may jeopardize those business relationships. The CFM MIB module separates objects by Maintenance Domain so that the bridge Owner can ensure that management by one Maintenance Domain administrator does not conflict with management by another.

Table 17-3—Sensitive managed objects: variables in dot1agCfmMdTable

Object	Reason for sensitivity to security considerations
`dot1agCfmMdFormat`, `dot1agCfmMdName`	An intentional or accidental write operation on these objects may lead to the modification of the identifiers of the Maintenance Domains and would impact the capacity of existing administrators to identify and manage the entities within those Maintenance Domains.
`dot1agCfmMdMdLevel`	An intentional or accidental write operation on this object may impact the CFM operations on the Maintenance Domain by altering the MD Level associated with the Maintenance Domain.
`dot1agCfmMdMhfCreation`	An intentional or accidental write operation on this object may impact the capabilities for creating MIP Half Functions (MHFs) in the Maintenance Domain.

17.5 Definitions for the Connectivity Fault Management MIB

In this subclause, certain terms (e.g., “`SHOULD`” and “`MUST`”) denote normative references according to the usage specified in IETF RFC 2119, as is customary for IETF MIB module definitions. In the following MIB definition, if any discrepancy between the DESCRIPTION text and the corresponding definition in any other part of this standard occurs, the definitions outside 17.5 take precedence.

NOTE—The following MIB module contains a textual convention, `Dot1agCfmPbbComponentIdentifier`, that is used to define indices into certain tables. These indices are required to accommodate the multiple bridge-like components expected to be defined in IEEE P802.1ah. They could also be used to configure additional CFM components in an IEEE 802.1ad Provider Bridge beyond those shown in Figure 22-10. The indices have been added to tables in this MIB module in the interest of not altering the MIB definition for IEEE Std 802.1ag-2007 soon after its completion. However, any corresponding changes to IEEE Std 802.1ad-2005 or IEEE Std 802.1ag-2007 required to introduce the concept of bridge components into Provider Bridges or CFM, respectively, if made, are expected be made as part of IEEE P802.1ah.

```
IEEE8021-CFM-MIB DEFINITIONS ::= BEGIN

-- ******************************************************************
-- IEEE P802.1ag(TM) CFM MIB
-- ******************************************************************

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    NOTIFICATION-TYPE,
    Integer32, Counter32,
    Unsigned32                FROM SNMPv2-SMI    -- [RFC2578]
    TEXTUAL-CONVENTION,
    TimeInterval,
    TimeStamp, RowStatus,
    TruthValue, MacAddress,
    TDomain, TAddress         FROM SNMPv2-TC     -- [RFC2579]
    MODULE-COMPLIANCE,
    OBJECT-GROUP,
    NOTIFICATION-GROUP        FROM SNMPv2-CONF   -- [RFC2580]
    InterfaceIndex,
    InterfaceIndexOrZero      FROM IF-MIB        -- [RFC2863]
    LldpChassisId,
    LldpChassisIdSubtype,
    LldpPortId,
    LldpPortIdSubtype         FROM LLDP-MIB      -- [IEEExxx]
    VlanIdOrNone, VlanId      FROM Q-BRIDGE-MIB  -- [RFC4363]
   ;

ieee8021CfmMib   MODULE-IDENTITY
    LAST-UPDATED "200705300000Z"    -- 05/30/2007 00:00GMT
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
       "WG-URL:   http://grouper.ieee.org/groups/802/1/index.html
        WG-EMail: stds-802-1@ieee.org

        Contact:  David Elie-Dit-Cosaque

                  Alcatel-Lucent
                  3400 W. Plano Pkwy.
                  Plano, TX 75075, USA

        E-mail:   david.elie_dit_cosaque@alcatel-lucent.com

        Contact:  Norman Finn

                  Cisco Systems
                  170 W. Tasman Drive
                  San Jose, CA 95134, USA

        E-mail:   nfinn@cisco.com
       "
   DESCRIPTION
     "Connectivity Fault Management module for managing IEEE 802.1ag"
   REVISION       "200705300000Z"    -- 05/30/2007 00:00GMT
   DESCRIPTION
      "Included in IEEE P802.1ag Draft 8.1

        Copyright (C) IEEE802.1."
    ::= { iso(1) org(3) ieee(111)
         standards-association-numbered-series-standards (2)
         lan-man-stds (802) ieee802dot1 (1) ieee802dot1mibs (1) 8 }

dot1agNotifications      OBJECT IDENTIFIER ::= { ieee8021CfmMib 0 }
dot1agMIBObjects         OBJECT IDENTIFIER ::= { ieee8021CfmMib 1 }
dot1agCfmConformance     OBJECT IDENTIFIER ::= { ieee8021CfmMib 2 }

-- ******************************************************************
-- Groups in the CFM MIB Module
-- ******************************************************************
dot1agCfmStack           OBJECT IDENTIFIER ::= { dot1agMIBObjects 1 }
dot1agCfmDefaultMd       OBJECT IDENTIFIER ::= { dot1agMIBObjects 2 }
dot1agCfmVlan            OBJECT IDENTIFIER ::= { dot1agMIBObjects 3 }
dot1agCfmConfigErrorList OBJECT IDENTIFIER ::= { dot1agMIBObjects 4 }
dot1agCfmMd              OBJECT IDENTIFIER ::= { dot1agMIBObjects 5 }
dot1agCfmMa              OBJECT IDENTIFIER ::= { dot1agMIBObjects 6 }
dot1agCfmMep             OBJECT IDENTIFIER ::= { dot1agMIBObjects 7 }

-- ******************************************************************
-- Textual conventions
-- ******************************************************************

Dot1agCfmMaintDomainNameType ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "A value that represents a type (and thereby the format)
        of a Dot1agCfmMaintDomainName.  The value can be one of
        the following:


        ieeeReserved(0)   Reserved for definition by IEEE 802.1
                          recommend to not use zero unless
                          absolutely needed.
        none(1)           No format specified, usually because
                          there is not (yet) a Maintenance
                          Domain Name. In this case, a zero
                          length OCTET STRING for the Domain
                          Name field is acceptable.
        dnsLikeName(2)    Domain Name like string, globally unique
                          text string derived from a DNS name.
        macAddrAndUint(3) MAC address + 2-octet (unsigned) integer.
        charString(4)     RFC2579 DisplayString, except that the
                          character codes 0-31 (decimal) are not
                          used.
        ieeeReserved(xx)  Reserved for definition by IEEE 802.1
                          xx values can be [5..31] and [64..255]
        ituReserved(xx)   Reserved for definition by  ITU-T Y.1731
                          xx values range from [32..63]
        To support future extensions, the Dot1agCfmMaintDomainNameType
        textual convention SHOULD NOT be sub-typed in object type
        definitions.  It MAY be sub-typed in compliance statements in
        order to require only a subset of these address types for a
        compliant implementation.

        Implementations must ensure that Dot1agCfmMaintDomainNameType
        objects and any dependent objects (e.g.,
        Dot1agCfmMaintDomainName objects) are consistent.  An
        inconsistentValue error must be generated if an attempt to
        change an Dot1agCfmMaintDomainNameType object would, for
        example, lead to an undefined Dot1agCfmMaintDomainName value.
        In particular,
        Dot1agCfmMaintDomainNameType/Dot1agCfmMaintDomainName pairs
        must be changed together if the nameType changes.
       "
    REFERENCE
       "802.1ag clause 21.6.5, Table 21-19"
    SYNTAX      INTEGER {
                  none              (1),
                  dnsLikeName       (2),
                  macAddressAndUint (3),
                  charString        (4)
                }

Dot1agCfmMaintDomainName ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Denotes a generic Maintenance Domain Name.

        A Dot1agCfmMaintDomainName value is always interpreted within
        the context of a Dot1agCfmMaintDomainNameType value.  Every
        usage of the Dot1agCfmMaintDomainName textual convention is
        required to specify the Dot1agCfmMaintDomainNameType object
        that provides the context.  It is suggested that the
        Dot1agCfmMaintDomainNameType object be logically registered
        before the object(s) that use the Dot1agCfmMaintDomainName
        textual convention, if they appear in the same logical row.

        The value of a Dot1agCfmMaintDomainName object must always
        be consistent with the value of the associated
        Dot1agCfmMaintDomainNameType object. Attempts to set
        an Dot1agCfmMaintDomainName object to a value inconsistent
        with the associated Dot1agCfmMaintDomainNameType must fail
        with an inconsistentValue error.

        When this textual convention is used as the syntax of an
        index object, there may be issues with the limit of 128
        sub-identifiers specified in SMIv2, IETF STD 58.  In this
        case, the object definition MUST include a 'SIZE' clause
        to limit the number of potential instance sub-identifiers;
        otherwise the applicable constraints MUST be stated in
        the appropriate conceptual row DESCRIPTION clauses, or
        in the surrounding documentation if there is no single
        DESCRIPTION clause that is appropriate.

        A value of none(1) in the associated
        Dot1agCfmMaintDomainNameType object means that no Maintenance
        Domain name is present, and the contents of the
        Dot1agCfmMaintDomainName object are meaningless.

        See the DESCRIPTION of the Dot1agCfmMaintAssocNameType
        TEXTUAL-CONVENTION for a discussion of the length limits on
        the Maintenance Domain name and Maintenance Association name.
       "
    REFERENCE
         "802.1ag clause 21.6.5"
    SYNTAX      OCTET STRING (SIZE(1..43))

Dot1agCfmMaintAssocNameType ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "A value that represents a type (and thereby the format)
        of a Dot1agCfmMaintAssocName.  The value can be one of
        the following:

        ieeeReserved(0)   Reserved for definition by IEEE 802.1
                          recommend to not use zero unless
                          absolutely needed.
        primaryVid(1)     Primary VLAN ID.
                          12 bits represented in a 2-octet integer:
                          - 4 least significant bits of the first
                            byte contains the 4 most significant
                            bits of the 12 bits primary VID
                          - second byte contains the 8 least
                            significant bits of the primary VID

                                 0 1 2 3 4 5 6 7 8
                                 +-+-+-+-+-+-+-+-+
                                 |0 0 0 0| (MSB) |
                                 +-+-+-+-+-+-+-+-+
                                 |  VID   LSB    |
                                 +-+-+-+-+-+-+-+-+

        charString(2)     RFC2579 DisplayString, except that the
                          character codes 0-31 (decimal) are not
                          used. (1..45) octets
        unsignedInt16 (3) 2-octet integer/big endian
        rfc2865VpnId(4)   RFC 2685 VPN ID
                          3 octet VPN authority Organizationally
                          Unique Identifier followed by 4 octet VPN
                          index identifying VPN according to the OUI:

                                 0 1 2 3 4 5 6 7 8
                                 +-+-+-+-+-+-+-+-+
                                 | VPN OUI (MSB) |
                                 +-+-+-+-+-+-+-+-+
                                 |   VPN OUI     |
                                 +-+-+-+-+-+-+-+-+
                                 | VPN OUI (LSB) |
                                 +-+-+-+-+-+-+-+-+
                                 |VPN Index (MSB)|
                                 +-+-+-+-+-+-+-+-+
                                 |  VPN Index    |
                                 +-+-+-+-+-+-+-+-+
                                 |  VPN Index    |
                                 +-+-+-+-+-+-+-+-+
                                 |VPN Index (LSB)|
                                 +-+-+-+-+-+-+-+-+

        ieeeReserved(xx)  Reserved for definition by IEEE 802.1
                          xx values can be [5..31] and [64..255]
        ituReserved(xx)   Reserved for definition by  ITU-T Y.1731
                          xx values range from [32..63]

        To support future extensions, the Dot1agCfmMaintAssocNameType
        textual convention SHOULD NOT be sub-typed in object type
        definitions.  It MAY be sub-typed in compliance statements in
        order to require only a subset of these address types for a
        compliant implementation.

        Implementations must ensure that Dot1agCfmMaintAssocNameType
        objects and any dependent objects (e.g.,
        Dot1agCfmMaintAssocName objects) are consistent.  An
        inconsistentValue error must be generated if an attempt to
        change an Dot1agCfmMaintAssocNameType object would, for
        example, lead to an undefined Dot1agCfmMaintAssocName value.
        In particular,
        Dot1agCfmMaintAssocNameType/Dot1agCfmMaintAssocName pairs
        must be changed together if the nameType changes.

        The Maintenance Domain name and Maintenance Association name,
        when put together into the CCM PDU, MUST total 48 octets or
        less.  If the Dot1agCfmMaintDomainNameType object contains
        none(1), then the Dot1agCfmMaintAssocName object MUST be
        45 octets or less in length.  Otherwise, the length of
        the Dot1agCfmMaintDomainName object plus the length of the
        Dot1agCfmMaintAssocName object, added together, MUST total
        less than or equal to 44 octets.
       "
    REFERENCE
       "802.1ag clause 21.6.5.4, Table 21-20"
    SYNTAX      INTEGER {
                  primaryVid        (1),
                  charString        (2),
                  unsignedInt16     (3),
                  rfc2865VpnId      (4)
                }

Dot1agCfmMaintAssocName ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Denotes a generic Maintenance Association Name. It is the
        part of the Maintenance Association Identifier which is
        unique within the Maintenance Domain Name and is appended
        to the Maintenance Domain Name to form the Maintenance
        Association Identifier (MAID).

        A Dot1agCfmMaintAssocName value is always interpreted within
        the context of a Dot1agCfmMaintAssocNameType value.  Every
        usage of the Dot1agCfmMaintAssocName textual convention is
        required to specify the Dot1agCfmMaintAssocNameType object
        that provides the context.  It is suggested that the
        Dot1agCfmMaintAssocNameType object be logically registered
        before the object(s) that use the Dot1agCfmMaintAssocName
        textual convention, if they appear in the same logical row.

        The value of a Dot1agCfmMaintAssocName object must
        always be consistent with the value of the associated
        Dot1agCfmMaintAssocNameType object. Attempts to set
        an Dot1agCfmMaintAssocName object to a value inconsistent
        with the associated Dot1agCfmMaintAssocNameType must fail
        with an inconsistentValue error.

        When this textual convention is used as the syntax of an
        index object, there may be issues with the limit of 128
        sub-identifiers specified in SMIv2, IETF STD 58.  In this
        case, the object definition MUST include a 'SIZE' clause
        to limit the number of potential instance sub-identifiers;
        otherwise the applicable constraints MUST be stated in
        the appropriate conceptual row DESCRIPTION clauses, or
        in the surrounding documentation if there is no single
        DESCRIPTION clause that is appropriate.
       "
    REFERENCE
       "802.1ag clauses 21.6.5.4, 21.6.5.5, 21.6.5.6"
    SYNTAX      OCTET STRING (SIZE(1..45))


Dot1agCfmMDLevel ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
       "Integer identifying the Maintenance Domain Level (MD Level).
        Higher numbers correspond to higher Maintenance Domains,
        those with the greatest physical reach, with the highest
        values for customers' CFM PDUs.  Lower numbers correspond
        to lower Maintenance Domains, those with more limited
        physical reach, with the lowest values for CFM PDUs
        protecting single bridges or physical links.
       "
    REFERENCE
       "802.1ag clauses 18.3, 21.4.1"
    SYNTAX      Integer32 (0..7)
```

```
Dot1agCfmMDLevelOrNone ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
       "Integer identifying the Maintenance Domain Level (MD Level).
        Higher numbers correspond to higher Maintenance Domains,
        those with the greatest physical reach, with the highest
        values for customers' CFM packets.  Lower numbers correspond
        to lower Maintenance Domains, those with more limited
        physical reach, with the lowest values for CFM PDUs
        protecting single bridges or physical links.

        The value (-1) is reserved to indicate that no MA Level has
        been assigned.
       "
    REFERENCE
       "802.1ag clauses 18.3, 12.14.3.1.3:c"
    SYNTAX      Integer32 (-1 | 0..7)

Dot1agCfmMpDirection ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Indicates the direction in which the Maintenance
        association (MEP or MIP) faces on the bridge port:

        down(1)    Sends Continuity Check Messages away from the
                   MAC Relay Entity.
        up(2)      Sends Continuity Check Messages towards the
                   MAC Relay Entity.
       "
    REFERENCE
       "802.1ag clauses 12.14.6.3.2:c"
    SYNTAX      INTEGER {
                  down  (1),
                  up    (2)
                }

Dot1agCfmPortStatus ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "An enumerated value from he Port Status TLV from the last CCM
        received from the last MEP. It indicates the ability of the
        Bridge Port on which the transmitting MEP resides to pass
        ordinary data, regardless of the status of the MAC
        (Table 21-10).

        psNoPortStateTLV(0) Indicates either that no CCM has been
                            received or that no port status TLV was
                            present in the last CCM received.

        psBlocked(1)        Ordinary data cannot pass freely through
                            the port on which the remote MEP resides.
                            Value of enableRmepDefect is equal to
                            false.
        psUp(2):            Ordinary data can pass freely through
                            the port on which the remote MEP resides.
                            Value of enableRmepDefect is equal to
                            true.

        NOTE: A 0 value is used for psNoPortStateTLV, so that
              additional code points can be added in a manner
              consistent with the Dot1agCfmInterfaceStatus textual
              convention.
       "
    REFERENCE
       "802.1ag clause 12.14.7.6.3:f, 20.19.3 and 21.5.4"
    SYNTAX      INTEGER {
                  psNoPortStateTLV  (0),
                  psBlocked         (1),
                  psUp              (2)
                }

Dot1agCfmInterfaceStatus ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "An enumerated value from the Interface Status TLV from the
        last CCM received from the last MEP. It indicates the status
        of the Interface within which the MEP transmitting the CCM
        is configured, or the next lower Interface in the Interface
        Stack, if the MEP is not configured within an Interface.

    isNoInterfaceStatusTLV(0)  Indicates either that no CCM has been
                           received or that no interface status TLV
                           was present in the last CCM received.

    isUp(1)                The interface is ready to pass packets.

    isDown(2)              The interface cannot pass packets

    isTesting(3)           The interface is in some test mode.

    isUnknown(4)           The interface status cannot be determined
                           for some reason.

    isDormant(5)           The interface is not in a state to pass
                           packets but is in a pending state, waiting
                           for some external event.

    isNotPresent(6)        Some component of the interface is missing

    isLowerLayerDown(7)    The interface is down due to state of the
                           lower layer interfaces

        NOTE: A 0 value is used for isNoInterfaceStatusTLV, so that
              these code points can be kept consistent with new code
              points added to ifOperStatus in the IF-MIB.
       "
    REFERENCE
       "802.1ag clause 12.14.7.6.3:g, 20.19.4 and 21.5.5"
    SYNTAX      INTEGER {
                  isNoInterfaceStatusTLV  (0),
                  isUp                    (1),
                  isDown                  (2),
                  isTesting               (3),
                  isUnknown               (4),
                  isDormant               (5),
                  isNotPresent            (6),
                  isLowerLayerDown        (7)
                }

Dot1agCfmHighestDefectPri ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "An enumerated value, equal to the contents of the variable
        highestDefect (20.33.9 and Table 20-1), indicating the
        highest-priority defect that has been present since the MEP
        Fault Notification Generator State Machine was last in the
        FNG_RESET state, either:

        none(0)           no defects since FNG_RESET
        defRDICCM(1)      DefRDICCM
        defMACstatus(2)   DefMACstatus
        defRemoteCCM(3)   DefRemoteCCM
        defErrorCCM(4)    DefErrorCCM
        defXconCCM(5)     DefXconCCM

        The value 0 is used for no defects so that additional higher
        priority values can be added, if needed, at a later time, and
        so that these values correspond with those in
        Dot1agCfmLowestAlarmPri.
       "
    REFERENCE
       "802.1ag clause 20.1.2, 12.14.7.7.2:c and 20.33.9"
    SYNTAX      INTEGER {
                  none              (0),
                  defRDICCM         (1),
                  defMACstatus      (2),
                  defRemoteCCM      (3),
                  defErrorCCM       (4),
                  defXconCCM        (5)
                }

Dot1agCfmLowestAlarmPri ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "An integer value specifying the lowest priority defect
        that is allowed to generate a Fault Alarm (20.9.5), either:

        allDef(1)           DefRDICCM, DefMACstatus, DefRemoteCCM,
                            DefErrorCCM, and DefXconCCM;
        macRemErrXcon(2)    Only DefMACstatus, DefRemoteCCM,
                            DefErrorCCM, and DefXconCCM (default);
        remErrXcon(3)       Only DefRemoteCCM, DefErrorCCM,
                            and DefXconCCM;
        errXcon(4)          Only DefErrorCCM and DefXconCCM;
        xcon(5)             Only DefXconCCM; or
        noXcon(6)           No defects DefXcon or lower are to be
                            reported;
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:k and 20.9.5"
    SYNTAX      INTEGER {
                  allDef            (1),
                  macRemErrXcon     (2),
                  remErrXcon        (3),
                  errXcon           (4),
                  xcon              (5),
                  noXcon            (6)
                }

Dot1agCfmMepId ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
       "Maintenance association End Point Identifier (MEPID): A small
        integer, unique over a given Maintenance Association,
        identifying a specific MEP.
       "
    REFERENCE
       "802.1ag clauses 3.19 and 19.2.1"
    SYNTAX      Unsigned32 (1..8191)

Dot1agCfmMepIdOrZero ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS       current
    DESCRIPTION
       "Maintenance association End Point Identifier (MEPID): A small
        integer, unique over a given Maintenance Association,
        identifying a specific MEP.

        The special value 0 is allowed to indicate special cases, for
        example that no MEPID is configured.

        Whenever an object is defined with this SYNTAX, then the
        DESCRIPTION clause of such an object MUST specify what the
        special value of 0 means.
       "
    REFERENCE
       "802.1ag clause 19.2.1"
    SYNTAX      Unsigned32 (0 | 1..8191)

Dot1agCfmMhfCreation ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
```

```
       "Indicates if the Management Entity can create MHFs.
        The valid values are:

        defMHFnone(1)      No MHFs can be created for this VID.
        defMHFdefault(2)   MHFs can be created on this VID on any
                           Bridge port through which this VID can
                           pass.
        defMHFexplicit(3)  MHFs can be created for this VID only on
                           Bridge ports through which this VID can
                           pass, and only if a MEP is created at some
                           lower MD Level.
        defMHFdefer(4)     The creation of MHFs is determined by the
                           corresponding Maintenance Domain variable
                           (dot1agCfmMaCompMhfCreation).
       "
    REFERENCE
       "802.1ag clause 12.14.5.1.3:c and 22.2.3"
    SYNTAX      INTEGER {
                  defMHFnone     (1),
                  defMHFdefault  (2),
                  defMHFexplicit (3),
                  defMHFdefer    (4)
                }

Dot1agCfmIdPermission ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Indicates what, if anything, is to be included in the Sender
        ID TLV transmitted in CCMs, LBMs, LTMs, and LTRs.  The valid
        values are:

        sendIdNone(1)      The Sender ID TLV is not to be sent.
        sendIdChassis(2)   The Chassis ID Length, Chassis ID
                           Subtype, and Chassis ID fields of  the
                           Sender ID TLV are to be sent.
        sendIdManage(3)    The Management Address Length and
                           Management Address of the Sender ID TLV
                           are to be sent.
        sendIdChassisManage(4) The Chassis ID Length, Chassis ID
                           Subtype, Chassis ID, Management Address
                           Length and Management Address fields are
                           all to be sent.
        sendIdDefer(5)     The contents of the Sender ID TLV are
                           determined by the corresponding
                           Maintenance Domain variable
                           (dot1agCfmMaCompIdPermission).
       "
    REFERENCE
       "802.1ag clause 12.14.6.1.3:d and 21.5.3"
    SYNTAX      INTEGER {
                  sendIdNone          (1),
                  sendIdChassis       (2),
                  sendIdManage        (3),
                  sendIdChassisManage (4),
                  sendIdDefer         (5)
                }

Dot1agCfmCcmInterval ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Indicates the interval at which CCMs are sent by a MEP.
        The possible values are:
        intervalInvalid(0) No CCMs are sent (disabled).
        interval300Hz(1)   CCMs are sent every 3 1/3 milliseconds
                           (300Hz).
        interval10ms(2)    CCMs are sent every 10 milliseconds.
        interval100ms(3)   CCMs are sent every 100 milliseconds.
        interval1s(4)      CCMs are sent every 1 second.
        interval10s(5)     CCMs are sent every 10 seconds.
        interval1min(6)    CCMs are sent every minute.
        interval10min(7)   CCMs are sent every 10 minutes.

        Note: enumerations start at zero to match the 'CCM Interval
              field' protocol field.
       "
    REFERENCE
       "802.1ag clauses 12.14.6.1.3:e, 20.8.1 and 21.6.1.3"
    SYNTAX      INTEGER {
                  intervalInvalid   (0),
                  interval300Hz     (1),
                  interval10ms      (2),
                  interval100ms     (3),
                  interval1s        (4),
                  interval10s       (5),
                  interval1min      (6),
                  interval10min     (7)
                }

Dot1agCfmFngState ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Indicates the diferent states of the MEP Fault Notification
        Generator State Machine.

        fngReset(1)            No defect has been present since the
                               dot1agCfmMepFngResetTime timer
                               expired, or since the state machine
                               was last reset.

        fngDefect(2)           A defect is present, but not for a
                               long enough time to be reported
                               (dot1agCfmMepFngAlarmTime).

        fngReportDefect(3)     A momentary state during which the
                               defect is reported by sending a
                               dot1agCfmFaultAlarm notification,
                               if that action is enabled.
```

```
         fngDefectReported(4)   A defect is present, and some defect
                                has been reported.

         fngDefectClearing(5)   No defect is present, but the
                                dot1agCfmMepFngResetTime timer has
                                not yet expired.
        "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:f and 20.35"
    SYNTAX      INTEGER {
                  fngReset          (1),
                  fngDefect         (2),
                  fngReportDefect   (3),
                  fngDefectReported (4),
                  fngDefectClearing (5)
                }

Dot1agCfmRelayActionFieldValue ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Possible values the Relay action field can take."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:g, 20.36.2.5, 21.9.5, and
        Table 21-27"
    SYNTAX      INTEGER {
                  rlyHit     (1),
                  rlyFdb     (2),
                  rlyMpdb    (3)
                }

Dot1agCfmIngressActionFieldValue ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Possible values returned in the ingress action field."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:g, 20.36.2.6, 21.9.8.1, and
        Table 21-30
       "
    SYNTAX      INTEGER {
                  ingNoTlv    (0),
                  ingOk       (1),
                  ingDown     (2),
                  ingBlocked  (3),
                  ingVid      (4)
                }

Dot1agCfmEgressActionFieldValue ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Possible values returned in the egress action field"
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:o, 20.36.2.10, 21.9.9.1, and
        Table 21-32"
    SYNTAX      INTEGER {
```

```
                  egrNoTlv    (0),
                  egrOK       (1),
                  egrDown     (2),
                  egrBlocked  (3),
                  egrVid      (4)
                }

Dot1agCfmRemoteMepState::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "Operational state of the remote MEP state machine.  This
        state machine monitors the reception of valid CCMs from a
        remote MEP with a specific MEPID.  It uses a timer that
        expires in 3.5 times the length of time indicated by the
        dot1agCfmMaNetCcmInterval object.

        rMepIdle(1)            Momentary state during reset.

        rMepStart(2)           The timer has not expired since the
                               state machine was reset, and no valid
                               CCM has yet been received.

        rMepFailed(3)          The timer has expired, both since the
                               state machine was reset, and since a
                               valid CCM was received.

        rMepOk(4)              The timer has not expired since a
                               valid CCM was received.
"
    REFERENCE
       "802.1ag clauses 12.14.7.6.3:b, 20.22"
    SYNTAX      INTEGER {
                  rMepIdle    (1),
                  rMepStart   (2),
                  rMepFailed  (3),
                  rMepOk      (4)
                }

Dot1afCfmIndexIntegerNextFree ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
      "An integer which may be used as a new Index in a table.

       The special value of 0 indicates that no more new entries can
       be created in the relevant table.

       When a MIB is used for configuration, an object with this
       SYNTAX always contains a legal value (if non-zero) for an
       index that is not currently used in the relevant table. The
       Command Generator (Network Management Application) reads this
       variable and uses the (non-zero) value read when creating a
       new row with an SNMP SET.  When the SET is performed, the
       Command Responder (agent) must determine whether the value is
       indeed still unused; Two Network Management Applications may
       attempt to create a row (configuration entry) simultaneously
       and use the same value. If it is currently unused, the SET
       succeeds and the Command Responder (agent) changes the value
       of this object, according to an implementation-specific
       algorithm.  If the value is in use, however, the SET fails.
       The Network Management Application must then re-read this
       variable to obtain a new usable value.

       An OBJECT-TYPE definition using this SYNTAX MUST specify the
       relevant table for which the object is providing this
       functionality.
      "
    SYNTAX      Unsigned32 (0..4294967295)

Dot1agCfmMepDefects ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "A MEP can detect and report a number of defects, and multiple
        defects can be present at the same time. These defects are:

        bDefRDICCM(0) A remote MEP is reported the RDI bit in its
                     last CCM.
        bDefMACstatus(1) Either some remote MEP is reporting its
                     Interface Status TLV as not isUp, or all remote
                     MEPs are reporting a Port Status TLV that
                     contains some value other than psUp.
        bDefRemoteCCM(2) The MEP is not receiving valid CCMs from at
                     least one of the remote MEPs.
        bDefErrorCCM(3) The MEP has received at least one invalid CCM
                     whose CCM Interval has not yet timed out.
        bDefXconCCM(4) The MEP has received at least one CCM from
                     either another MAID or a lower MD Level whose
                     CCM Interval has not yet timed out.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:o, 12.14.7.1.3:p, 12.14.7.1.3:q,
        12.14.7.1.3:r, and 12.14.7.1.3:s."
    SYNTAX BITS {
                bDefRDICCM(0),
                bDefMACstatus(1),
                bDefRemoteCCM(2),
                bDefErrorCCM(3),
                bDefXconCCM(4)
               }

Dot1agCfmConfigErrors ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
       "While making the MIP creation evaluation described in 802.1ag
        clause 22.2.3, the management entity can encounter errors in
        the configuration. These are possible errors that can be
        encountered:
```

```
        CFMleak(0)   MA x is associated with a specific VID list,
                     one or more of the VIDs in MA x can pass through
                     the Bridge Port, no Down MEP is configured on
                     any Bridge Port for MA x, and some other MA y,
                     at a higher MD Level than MA x, and associated
                     with at least one of the VID(s) also in MA x,
                     does have a MEP configured on the Bridge Port.

        conflictingVids(1)  MA x is associated with a specific VID
                     list, an Up MEP is configured on MA x on the
                     Bridge Port, and some other MA y, associated
                     with at least one of the VID(s) also in MA x,
                     also has an Up MEP configured on some Bridge
                     Port.

        ExcessiveLevels(2)  The number of different MD Levels at
                     which MIPs are to be created on this port
                     exceeds the Bridge's capabilities (see
                     subclause 22.3).

        OverlappedLevels(3) A MEP is created for one VID at one MD
                     Level, but a MEP is configured on another
                     VID at that MD Level or higher, exceeding
                     the Bridge's capabilities.
       "
    REFERENCE
       "802.1ag clause 12.14.4.1.3:b and clauses 22.2.3 and 22.2.4"
    SYNTAX BITS {
                cfmLeak(0),
                conflictingVids(1),
                excessiveLevels(2),
                overlappedLevels(3)
                }

Dot1agCfmPbbComponentIdentifier ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
       "A Provider Backbone Bridge (PBB) can comprise a number of
        components, each of which can be managed in a manner
        essentially equivalent to an 802.1Q bridge.  In order to access
        these components easily, an index is used in a number of
        tables.  If any two tables are indexed by
        Dot1agCfmPbbComponentIdentifier, then entries in those tables
        indexed by the same value of Dot1agCfmPbbComponentIdentifier
        correspond to the same component.
       "
    REFERENCE
       "802.1ag clause 17.5"
    SYNTAX      Unsigned32 (1..4294967295)

-- *******************************************************************
-- The Stack Object. This group will contain all the MIBs objects
-- needed to access the Stack managed object.
-- ******************************************************************

-- ******************************************************************
-- The CFM Stack Table
-- ******************************************************************

dot1agCfmStackTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmStackEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "There is one CFM Stack table per bridge. It permits
        the retrieval of information about the Maintenance Points
        configured on any given interface.
       "
    REFERENCE
       "802.1ag clauses 12.14.2"
    ::= { dot1agCfmStack 1 }

dot1agCfmStackEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmStackEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Stack table entry"
    INDEX { dot1agCfmStackifIndex, dot1agCfmStackVlanIdOrNone,
            dot1agCfmStackMdLevel, dot1agCfmStackDirection
          }
    ::= { dot1agCfmStackTable 1 }

Dot1agCfmStackEntry ::= SEQUENCE {
      dot1agCfmStackifIndex       InterfaceIndex,
      dot1agCfmStackVlanIdOrNone  VlanIdOrNone,
      dot1agCfmStackMdLevel       Dot1agCfmMDLevel,
      dot1agCfmStackDirection     Dot1agCfmMpDirection,
      dot1agCfmStackMdIndex       Unsigned32,
      dot1agCfmStackMaIndex       Unsigned32,
      dot1agCfmStackMepId         Dot1agCfmMepIdOrZero,
      dot1agCfmStackMacAddress    MacAddress
    }

dot1agCfmStackifIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This object represents the  Bridge Port or aggregated port
        on which MEPs or MHFs might be configured.

        Upon a restart of the system, the system SHALL, if necessary,
        change the value of this variable, and  rearrange the
        dot1agCfmStackTable, so that it indexes the entry in the
        interface table with the same value of ifAlias that it
        indexed before the system restart.  If no such entry exists,
        then the system SHALL delete all entries in the
        dot1agCfmStackTable with the interface index.
       "
    REFERENCE
       "802.1ag clause 12.14.2.1.2:a"
    ::= { dot1agCfmStackEntry 1 }

dot1agCfmStackVlanIdOrNone OBJECT-TYPE
    SYNTAX      VlanIdOrNone
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "VLAN ID to which the MP is attached, or 0, if none."
    REFERENCE
       "802.1ag clauses 12.14.2.1.2:d, 22.1.7"
    ::= { dot1agCfmStackEntry 2 }

dot1agCfmStackMdLevel OBJECT-TYPE
    SYNTAX      Dot1agCfmMDLevel
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "MD Level of the Maintenance Point."
    REFERENCE
       "802.1ag clause 12.14.2.1.2:b"
    ::= { dot1agCfmStackEntry 3 }

dot1agCfmStackDirection OBJECT-TYPE
    SYNTAX      Dot1agCfmMpDirection
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Direction in which the MP faces on the Bridge Port"
    REFERENCE
       "802.1ag clause 12.14.2.1.2:c"
    ::= { dot1agCfmStackEntry 4 }

dot1agCfmStackMdIndex OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The index of the Maintenance Domain in the dot1agCfmMdTable
        to which the MP is associated, or 0, if none."
    REFERENCE
       "802.1ag clause 12.14.2.1.3:b"
    ::= { dot1agCfmStackEntry 5 }

dot1agCfmStackMaIndex OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The index of the MA in the dot1agCfmMaNetTable and
```

```
        dot1agCfmMaCompTable to which the MP is associated, or 0, if
        none."
    REFERENCE
       "802.1ag clause 12.14.2.1.3:c"
    ::= { dot1agCfmStackEntry 6 }

dot1agCfmStackMepId OBJECT-TYPE
    SYNTAX      Dot1agCfmMepIdOrZero
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "If an MEP is configured, the MEPID, else 0"
    REFERENCE
       "802.1ag clause 12.14.2.1.3:d"
    ::= { dot1agCfmStackEntry 7 }

dot1agCfmStackMacAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "MAC address of the MP."
    REFERENCE
      "802.1ag clause 12.14.2.1.3:e"
    ::= { dot1agCfmStackEntry 8 }

-- ******************************************************************
-- The VLAN Table
-- ******************************************************************

dot1agCfmVlanTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This table defines the association of VIDs into VLANs.  There
        is an entry in this table, for each component of the bridge,
        for each VID that is:
            a) a VID belonging to a VLAN associated with more than
               one VID; and
            b) not the Primary VLAN of that VID.
        The entry in this table contains the Primary VID of the VLAN.

        By default, this table is empty, meaning that every VID is
        the Primary VID of a single-VID VLAN.

        VLANs that are associated with only one VID SHOULD NOT have
        an entry in this table.

        The writable objects in this table need to be persistent
        upon reboot or restart of a device.
       "
    REFERENCE
       "802.1ag clauses 12.14.3.1.3:a, 12.14.3.2.2:a, 12.14.5.3.2:c,
        12.14.6.1.3:b, 22.1.5."
    ::= { dot1agCfmVlan 1 }

dot1agCfmVlanEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The VLAN table entry."
    INDEX { dot1agCfmVlanComponentId, dot1agCfmVlanVid }
    ::= { dot1agCfmVlanTable 1 }

Dot1agCfmVlanEntry ::= SEQUENCE {
      dot1agCfmVlanComponentId Dot1agCfmPbbComponentIdentifier,
      dot1agCfmVlanVid         VlanId,
      dot1agCfmVlanPrimaryVid  VlanId,
      dot1agCfmVlanRowStatus   RowStatus
    }

dot1agCfmVlanComponentId OBJECT-TYPE
    SYNTAX      Dot1agCfmPbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The bridge component within the system to which the information
        in this dot1agCfmVlanEntry applies.  If the system is not a
        Bridge, or if only one component is present in the Bridge, then
        this variable (index) must be equal to 1.
       "
    REFERENCE
       "802.1ag clause 17.5"
    ::= { dot1agCfmVlanEntry 1 }

dot1agCfmVlanVid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This is a VLAN ID belonging to a VLAN that is associated with
        more than one VLAN ID, and this is not the Primary VID of the
        VLAN.
       "
    ::= { dot1agCfmVlanEntry 2 }

dot1agCfmVlanPrimaryVid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "This is the Primary VLAN ID of the VLAN with which this
        entry's dot1agCfmVlanVid is associated.  This value must not
        equal the value of dot1agCfmVlanVid.
       "
    ::= { dot1agCfmVlanEntry 3 }
```

```
dot1agCfmVlanRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The status of the row.

        The writable columns in a row can not be changed if the row
        is active. All columns must have a valid value before a row
        can be activated.
       "
    ::= { dot1agCfmVlanEntry 4 }

-- *******************************************************************
-- The Default MD Level object. This group will contain all the
-- MIB objects needed to access and modify default MD level
-- managed objects.
-- *******************************************************************

dot1agCfmDefaultMdDefLevel OBJECT-TYPE
    SYNTAX      Dot1agCfmMDLevel
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "A value indicating the MD Level at which MHFs are to be
        created, and Sender ID TLV transmission by those MHFs is to
        be controlled, for each dot1agCfmDefaultMdEntry whose
        dot1agCfmDefaultMdLevel object contains the value -1.

        After this initialization, this object needs to be persistent
        upon reboot or restart of a device.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:c, 12.14.3.2.2:b"
    DEFVAL {0}
    ::= { dot1agCfmDefaultMd 1 }

dot1agCfmDefaultMdDefMhfCreation OBJECT-TYPE
    SYNTAX      Dot1agCfmMhfCreation {
                  defMHFnone     (1),
                  defMHFdefault  (2),
                  defMHFexplicit (3)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "A value indicating if the Management entity can create MHFs
        (MIP Half Function) for the VID, for each
        dot1agCfmDefaultMdEntry whose dot1agCfmDefaultMdMhfCreation
        object contains the value defMHFdefer.  Since, in this
        variable, there is no encompassing Maintenance Domain, the
        value defMHFdefer is not allowed.
        After this initialization, this object needs to be persistent
        upon reboot or restart of a device.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:d"
    DEFVAL {defMHFnone}
    ::= { dot1agCfmDefaultMd 2 }

dot1agCfmDefaultMdDefIdPermission OBJECT-TYPE
    SYNTAX      Dot1agCfmIdPermission {
                  sendIdNone          (1),
                  sendIdChassis       (2),
                  sendIdManage        (3),
                  sendIdChassisManage (4)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "Enumerated value indicating what, if anything, is to be
        included in the Sender ID TLV (21.5.3) transmitted by MHFs
        created by the Default Maintenance Domain, for each
        dot1agCfmDefaultMdEntry whose dot1agCfmDefaultMdIdPermission
        object contains the value sendIdDefer.  Since, in this
        variable, there is no encompassing Maintenance Domain, the
        value sendIdDefer is not allowed.

        After this initialization, this object needs to be persistent
        upon reboot or restart of a device.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:e"
    DEFVAL { sendIdNone }
    ::= { dot1agCfmDefaultMd 3 }

-- ******************************************************************
-- The Default MD Level Table
-- ******************************************************************

dot1agCfmDefaultMdTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmDefaultMdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "For each bridge component, the Default MD Level Managed Object
        controls MHF creation for VIDs that are not attached to a
        specific Maintenance Association Managed Object, and Sender ID
        TLV transmission by those MHFs.

        For each Bridge Port, and for each VLAN ID whose data can
        pass through that Bridge Port, an entry in this table is
        used by the algorithm in subclause 22.2.3 only if there is no
        entry in the Maintenance Association table defining an MA
        for the same VLAN ID and MD Level as this table's entry, and
        on which MA an Up MEP is defined.  If there exists such an
```

```
         MA, that MA's objects are used by the algorithm in
         subclause 22.2.3 in place of this table entry's objects.  The
         agent maintains the value of dot1agCfmDefaultMdStatus to
         indicate whether this entry is overridden by an MA.

         When first initialized, the agent creates this table
         automatically with entries for all VLAN IDs,
         with the default values specified for each object.

         After this initialization, the writable objects in this
         table need to be persistent upon reboot or restart of a
         device.
        "
    REFERENCE
       "802.1ag clause 12.14.3"
    ::= { dot1agCfmDefaultMd 4 }

dot1agCfmDefaultMdEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmDefaultMdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Default MD Level table entry."
    INDEX { dot1agCfmDefaultMdComponentId,
            dot1agCfmDefaultMdPrimaryVid }
    ::= { dot1agCfmDefaultMdTable 1 }

Dot1agCfmDefaultMdEntry ::= SEQUENCE {
      dot1agCfmDefaultMdComponentId  Dot1agCfmPbbComponentIdentifier,
      dot1agCfmDefaultMdPrimaryVid   VlanId,
      dot1agCfmDefaultMdStatus       TruthValue,
      dot1agCfmDefaultMdLevel        Dot1agCfmMDLevelOrNone,
      dot1agCfmDefaultMdMhfCreation  Dot1agCfmMhfCreation,
      dot1agCfmDefaultMdIdPermission Dot1agCfmIdPermission
    }

dot1agCfmDefaultMdComponentId OBJECT-TYPE
    SYNTAX      Dot1agCfmPbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The bridge component within the system to which the information
        in this dot1agCfmDefaultMdEntry applies.  If the system is not
        a Bridge, or if only one component is present in the Bridge,
        then this variable (index) must be equal to 1.
       "
    REFERENCE
       "802.1ag clause 17.5"
    ::= { dot1agCfmDefaultMdEntry 1 }

dot1agCfmDefaultMdPrimaryVid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Primary VID of the VLAN to which this entry's objects
        apply."
    ::= { dot1agCfmDefaultMdEntry 2 }

dot1agCfmDefaultMdStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "State of this Default MD Level table entry.  True if there is
        no entry in the Maintenance Association table defining an MA
        for the same VLAN ID and MD Level as this table's entry, and
        on which MA an Up MEP is defined, else false.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:b"
    ::= { dot1agCfmDefaultMdEntry 3 }

dot1agCfmDefaultMdLevel OBJECT-TYPE
    SYNTAX      Dot1agCfmMDLevelOrNone
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "A value indicating the MD Level at which MHFs are to be
        created, and Sender ID TLV transmission by those MHFs is to
        be controlled, for the VLAN to which this entry's objects
        apply.  If this object has the value -1, the MD Level for MHF
        creation for this VLAN is controlled by
        dot1agCfmDefaultMdDefLevel.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:c, 12.14.3.2.2:b"
    DEFVAL {-1}
    ::= { dot1agCfmDefaultMdEntry 4 }

dot1agCfmDefaultMdMhfCreation OBJECT-TYPE
    SYNTAX      Dot1agCfmMhfCreation
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "A value indicating if the Management entity can create MHFs
        (MIP Half Function) for this VID at this MD Level.  If this
        object has the value defMHFdefer, MHF creation for this VLAN
        is controlled by dot1agCfmDefaultMdDefMhfCreation.

        The value of this variable is meaningless if the values of
        dot1agCfmDefaultMdStatus is false.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:d"
    DEFVAL {defMHFdefer}
    ::= { dot1agCfmDefaultMdEntry 5 }
```

```
dot1agCfmDefaultMdIdPermission OBJECT-TYPE
    SYNTAX      Dot1agCfmIdPermission
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "Enumerated value indicating what, if anything, is to be
        included in the Sender ID TLV (21.5.3) transmitted by MHFs
        created by the Default Maintenance Domain.  If this object
        has the value sendIdDefer, Sender ID TLV transmission for
        this VLAN is controlled by dot1agCfmDefaultMdDefIdPermission.

        The value of this variable is meaningless if the values of
        dot1agCfmDefaultMdStatus is false.
       "
    REFERENCE
       "802.1ag clause 12.14.3.1.3:e"
    DEFVAL { sendIdDefer }
    ::= { dot1agCfmDefaultMdEntry 6 }

-- ******************************************************************
-- The CFM configuration error list managed object. This group will
-- contain all the MIB objects used to read the interfaces and VIDs
-- configured incorrectly.
-- ******************************************************************

-- ******************************************************************
-- The CFM Configuration Error List Table
-- ******************************************************************

dot1agCfmConfigErrorListTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmConfigErrorListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The CFM Configuration Error List table provides a list of
        Interfaces and VIDs that are incorrectly configured.
       "
    REFERENCE
       "802.1ag clause 12.14.4"
    ::= {dot1agCfmConfigErrorList 1}

dot1agCfmConfigErrorListEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmConfigErrorListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Config Error List Table  entry"
    INDEX { dot1agCfmConfigErrorListVid,
            dot1agCfmConfigErrorListIfIndex
          }
    ::= { dot1agCfmConfigErrorListTable 1}

Dot1agCfmConfigErrorListEntry ::= SEQUENCE {
      dot1agCfmConfigErrorListVid         VlanId,
      dot1agCfmConfigErrorListIfIndex     InterfaceIndex,
      dot1agCfmConfigErrorListErrorType   Dot1agCfmConfigErrors
    }


dot1agCfmConfigErrorListVid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The VLAN ID of the VLAN with interfaces in error."
    REFERENCE
       "802.1ag Clause 12.14.4.1.2:a"
    ::= { dot1agCfmConfigErrorListEntry 1 }

dot1agCfmConfigErrorListIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This object is the IfIndex of the interface.

        Upon a restart of the system, the system SHALL, if necessary,
        change the value of this variable so that it indexes the
        entry in the interface table with the same value of ifAlias
        that it indexed before the system restart.  If no such
        entry exists, then the system SHALL delete any entries in
        dot1agCfmConfigErrorListTable indexed by that
        InterfaceIndex value.
       "
    REFERENCE
       "802.1ag clause 12.14.4.1.2:b"
    ::= { dot1agCfmConfigErrorListEntry 2 }

dot1agCfmConfigErrorListErrorType OBJECT-TYPE
    SYNTAX      Dot1agCfmConfigErrors
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "A vector of Boolean error conditions from 22.2.4, any of
        which may be true:

        0) CFMleak;
        1) ConflictingVids;
        2) ExcessiveLevels;
        3) OverlappedLevels.
       "
    REFERENCE
       "802.1ag clause 12.14.4.1.3:b"
    ::= { dot1agCfmConfigErrorListEntry 3 }

-- *****************************************************************
-- The Maintenance Domain Managed Object.  This group contains all
-- the MIB objects used to maintain Maintenance Domains.
-- ******************************************************************

dot1agCfmMdTableNextIndex OBJECT-TYPE
    SYNTAX      Dot1afCfmIndexIntegerNextFree
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "This object contains an unused value for dot1agCfmMdIndex in
        the dot1agCfmMdTable, or a zero to indicate that none exist.
       "
    ::= { dot1agCfmMd 1 }

-- ******************************************************************
-- The Maintenance Domain Table
-- ******************************************************************

dot1agCfmMdTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmMdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Maintenance Domain table. Each row in the table
        represents a different Maintenance Domain.

        A Maintenance Domain is described in 802.1ag (3.22) as the
        network or the part of the network for which faults in
        connectivity are to be managed. The boundary of a Maintenance
        Domain is defined by a set of DSAPs, each of which can become
        a point of connectivity to a service instance.
       "
    REFERENCE
       "802.1ag clauses 3.22 and 18.1"
    ::= { dot1agCfmMd 2 }

dot1agCfmMdEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmMdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Maintenance Domain table entry. This entry is not lost
        upon reboot. It is backed up by stable storage.
       "
    INDEX {dot1agCfmMdIndex }
    ::= { dot1agCfmMdTable 1 }

Dot1agCfmMdEntry ::= SEQUENCE {
      dot1agCfmMdIndex                Unsigned32,
      dot1agCfmMdFormat               Dot1agCfmMaintDomainNameType,
      dot1agCfmMdName                 Dot1agCfmMaintDomainName,
      dot1agCfmMdMdLevel              Dot1agCfmMDLevel,
      dot1agCfmMdMhfCreation          Dot1agCfmMhfCreation,
      dot1agCfmMdMhfIdPermission      Dot1agCfmIdPermission,
      dot1agCfmMdMaNextIndex          Dot1afCfmIndexIntegerNextFree,
      dot1agCfmMdRowStatus            RowStatus
    }

dot1agCfmMdIndex OBJECT-TYPE
    SYNTAX      Unsigned32(1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The index to the Maintenance Domain table.

        dot1agCfmMdTableNextIndex needs to be inspected to find an
        available index for row-creation.

        Referential integrity is required, i.e., the index needs to be
        persistent upon a reboot or restart of a device.  The index
        can never be reused for other Maintenance Domain.  The index
        value should keep increasing up to the time that they wrap
        around. This is to facilitate access control based on OID.
       "
    ::= { dot1agCfmMdEntry 1 }

dot1agCfmMdFormat OBJECT-TYPE
    SYNTAX      Dot1agCfmMaintDomainNameType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The type (and thereby format) of the Maintenance Domain Name."
    REFERENCE
       "802.1ag clause 21.6.5.1"
    DEFVAL { charString }
    ::= { dot1agCfmMdEntry 2 }

dot1agCfmMdName OBJECT-TYPE
    SYNTAX      Dot1agCfmMaintDomainName
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Maintenance Domain name. The type/format of this object
        is determined by the value of the dot1agCfmMdNameType object.

        Each Maintenance Domain has unique name amongst all those
        used or available to a service provider or operator.  It
        facilitates easy identification of administrative
        responsibility for each Maintenance Domain.

        Clause 3.24 defines a Maintenance Domain name as the
        identifier, unique over the domain for which CFM is to
        protect against accidental concatenation of Service
        Instances, of a particular Maintenance Domain.
       "
    REFERENCE
       "802.1ag clauses 3.24, 12.14.5, and 21.6.5.3"
    DEFVAL { "DEFAULT" }
    ::= { dot1agCfmMdEntry 3 }
```

```
dot1agCfmMdMdLevel OBJECT-TYPE
    SYNTAX      Dot1agCfmMDLevel
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Maintenance Domain Level."
    REFERENCE
       "802.1ag clause 12.14.5.1.3:b"
    DEFVAL { 0 }
    ::= { dot1agCfmMdEntry 4 }

dot1agCfmMdMhfCreation OBJECT-TYPE
    SYNTAX      Dot1agCfmMhfCreation {
                  defMHFnone     (1),
                  defMHFdefault  (2),
                  defMHFexplicit (3)
                }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Enumerated value indicating whether the management entity can
        create MHFs (MIP Half Function) for this Maintenance Domain.
        Since, in this variable, there is no encompassing Maintenance
        Domain, the value defMHFdefer is not allowed.
       "
    REFERENCE
       "802.1ag clause 12.14.5.1.3:c"
    DEFVAL { defMHFnone }
    ::= { dot1agCfmMdEntry 5 }

dot1agCfmMdMhfIdPermission OBJECT-TYPE
    SYNTAX      Dot1agCfmIdPermission {
                  sendIdNone          (1),
                  sendIdChassis       (2),
                  sendIdManage        (3),
                  sendIdChassisManage (4)
                }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Enumerated value indicating what, if anything, is to be
        included in the Sender ID TLV (21.5.3) transmitted by MPs
        configured in this Maintenance Domain.  Since, in this
        variable, there is no encompassing Maintenance Domain, the
        value sendIdDefer is not allowed.
       "
    REFERENCE
       "802.1ag clause 12.14.5.1.3:d"
    DEFVAL { sendIdNone }
    ::= { dot1agCfmMdEntry 6 }

dot1agCfmMdMaNextIndex OBJECT-TYPE
    SYNTAX      Dot1afCfmIndexIntegerNextFree
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Value to be used as the index of the MA table entries, both
        the dot1agCfmMaNetTable and the dot1agCfmMaCompTable, for
        this Maintenance Domain when the management entity wants to
        create a new row in those tables.
       "
   ::= { dot1agCfmMdEntry 7 }

dot1agCfmMdRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The status of the row.

        The writable columns in a row can not be changed if the row
        is active. All columns must have a valid value before a row
        can be activated.
       "
    ::= { dot1agCfmMdEntry 8 }

-- *****************************************************************
-- The Maintenance Association Object. This group contains all the
-- MIB objects used to read, create, modify, and delete Maintenance
-- Associations in the MIB.
-- *****************************************************************

-- *****************************************************************
-- The Maintenance Association (MA) Network Table
-- *****************************************************************

dot1agCfmMaNetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmMaNetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Maintenance Association table.  Each row in the table
        represents an MA.  An MA is a set of MEPs, each configured
        with a single service instance.

        This is the part of the complete MA table that is constant
        across all Bridges in a Maintenance Domain, and across all
        components of a single Bridge.  That part of the MA table that
        can vary from Bridge component to Bridge component is contained
        in the dot1agCfmMaCompTable.

        Creation of a Service Instance establishes a connectionless
        association among the selected DSAPs.  Configuring a
        Maintenance association End Point (MEP) at each of the
        DSAPs creates a Maintenance Association (MA) to monitor
        that connectionless connectivity.  The MA is identified by a
        Short MA Name that is unique within the Maintenance Domain
        and chosen to facilitate easy identification of the Service
        Instance.  Together, the Maintenance Domain Name and the
        Short MA Name form the Maintenance Association Identifier
        (MAID) that is carried in CFM Messages to identify
        incorrect connectivity among Service Instances.  A small
        integer, the Maintenance association End Point Identifier
        (MEPID), identifies each MEP among those configured on a
        single MA (802.1ag clauses 3.17 and 18.2).

        This table uses two indices, first index is the index of the
        Maintenance Domain table.  The second index is the same as the
        index of the dot1agCfmMaCompEntry for the same MA.

        The writable objects in this table need to be persistent
        upon reboot or restart of a device.

       "
    REFERENCE
       "802.1ag clause 18.2"
    ::= { dot1agCfmMa 1 }

dot1agCfmMaNetEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmMaNetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The MA table entry."
    INDEX {dot1agCfmMdIndex, dot1agCfmMaIndex }
    ::= { dot1agCfmMaNetTable 1 }

Dot1agCfmMaNetEntry ::= SEQUENCE {
      dot1agCfmMaIndex                   Unsigned32,
      dot1agCfmMaNetFormat               Dot1agCfmMaintAssocNameType,
      dot1agCfmMaNetName                 Dot1agCfmMaintAssocName,
      dot1agCfmMaNetCcmInterval          Dot1agCfmCcmInterval,
      dot1agCfmMaNetRowStatus            RowStatus
    }

dot1agCfmMaIndex OBJECT-TYPE
    SYNTAX      Unsigned32(1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION

       "Index of the MA table dot1agCfmMdMaNextIndex needs to
        be inspected to find an available index for row-creation.
       "
    ::= { dot1agCfmMaNetEntry 1 }

dot1agCfmMaNetFormat OBJECT-TYPE
    SYNTAX      Dot1agCfmMaintAssocNameType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The type (and thereby format) of the Maintenance Association
```

```
         Name.
        "
    REFERENCE
       "802.1ag clauses 21.6.5.4"
    ::= { dot1agCfmMaNetEntry 2 }

dot1agCfmMaNetName OBJECT-TYPE
    SYNTAX      Dot1agCfmMaintAssocName
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Short Maintenance Association name. The type/format of
        this object is determined by the value of the
        dot1agCfmMaNetNameType object.  This name must be unique within
        a maintenance domain.
       "
    REFERENCE
       "802.1ag clauses 21.6.5.6, and Table 21-20"
    ::= { dot1agCfmMaNetEntry 3 }

dot1agCfmMaNetCcmInterval OBJECT-TYPE
    SYNTAX      Dot1agCfmCcmInterval
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Interval between CCM transmissions to be used by all MEPs
        in the MA.
       "
    REFERENCE
       "802.1ag clause 12.14.6.1.3:e"
    DEFVAL { interval1s }
    ::= { dot1agCfmMaNetEntry 4 }

dot1agCfmMaNetRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The status of the row.

        The writable columns in a row can not be changed if the row
        is active. All columns must have a valid value before a row
        can be activated.
       "
    ::= { dot1agCfmMaNetEntry 5 }

-- *****************************************************************
-- The Maintenance Association (MA) Component Table
-- *****************************************************************

dot1agCfmMaCompTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmMaCompEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Maintenance Association table.  Each row in the table
        represents an MA.  An MA is a set of MEPs, each configured
        with a single service instance.

        This is the part of the complete MA table that is variable
        across the Bridges in a Maintenance Domain, or across the
        components of a single Bridge.  That part of the MA table that
        is constant across the Bridges and their components in a
        Maintenance Domain is contained in the dot1agCfmMaNetTable.

        This table uses three indices, first index is the
        Dot1agCfmPbbComponentIdentifier that identifies the component
        within the Bridge for which the information in the
        dot1agCfmMaCompEntry applies.  The second is the index of the
        Maintenance Domain table.  The third index is the same as the
        index of the dot1agCfmMaNetEntry for the same MA.

        The writable objects in this table need to be persistent
        upon reboot or restart of a device.

       "
    REFERENCE
       "802.1ag clause 18.2"
    ::= { dot1agCfmMa 2 }

dot1agCfmMaCompEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmMaCompEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The MA table entry."
    INDEX {dot1agCfmMaComponentId,
           dot1agCfmMdIndex, dot1agCfmMaIndex }
    ::= { dot1agCfmMaCompTable 1 }

Dot1agCfmMaCompEntry ::= SEQUENCE {
      dot1agCfmMaComponentId         Dot1agCfmPbbComponentIdentifier,
      dot1agCfmMaCompPrimaryVlanId   VlanIdOrNone,
      dot1agCfmMaCompMhfCreation     Dot1agCfmMhfCreation,
      dot1agCfmMaCompIdPermission    Dot1agCfmIdPermission,
      dot1agCfmMaCompNumberOfVids    Unsigned32,
      dot1agCfmMaCompRowStatus       RowStatus
    }

dot1agCfmMaComponentId OBJECT-TYPE
    SYNTAX      Dot1agCfmPbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The bridge component within the system to which the information
        in this dot1agCfmMaCompEntry applies.  If the system is not a
        Bridge, or if only one component is present in the Bridge, then
        this variable (index) must be equal to 1.
       "
    REFERENCE
       "802.1ag clause 17.5"
    ::= { dot1agCfmMaCompEntry 1 }

dot1agCfmMaCompPrimaryVlanId OBJECT-TYPE
    SYNTAX      VlanIdOrNone
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Primary VLAN ID with which the Maintenance Association is
        associated, or 0 if the MA is not attached to any VID.  If
        the MA is associated with more than one VID, the
        dot1agCfmVlanTable lists them."
    REFERENCE
       "802.1ag clause 12.14.6.1.3:b"
    ::= { dot1agCfmMaCompEntry 2 }

dot1agCfmMaCompMhfCreation OBJECT-TYPE
    SYNTAX      Dot1agCfmMhfCreation
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Indicates if the Management entity can create MHFs (MIP Half
        Function) for this MA.
       "
    REFERENCE
       "802.1ag clause 12.14.6.1.3:c"
    DEFVAL { defMHFdefer }
    ::= { dot1agCfmMaCompEntry 3 }

dot1agCfmMaCompIdPermission OBJECT-TYPE
    SYNTAX      Dot1agCfmIdPermission
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Enumerated value indicating what, if anything, is to be
        included in the Sender ID TLV (21.5.3) transmitted by MPs
        configured in this MA.
       "
    REFERENCE
       "802.1ag clause 12.14.6.1.3:d"
    DEFVAL { sendIdDefer }
    ::= { dot1agCfmMaCompEntry 4 }

dot1agCfmMaCompNumberOfVids OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The number of VIDs associated with the MA.
       "
    REFERENCE
       "802.1ag clause 12.14.6.1.3:b"
    ::= { dot1agCfmMaCompEntry 5 }

dot1agCfmMaCompRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The status of the row.

        The writable columns in a row can not be changed if the row
        is active. All columns must have a valid value before a row
        can be activated.
       "
    ::= { dot1agCfmMaCompEntry 6 }

-- *******************************************************************
-- The list of known MEPs for a given MA
-- *******************************************************************

dot1agCfmMaMepListTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmMaMepListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "List of MEPIDs that belong to this MA.

        Clause 12.14.6.1.3 specifies that a list of MEPIDs in all
        bridges in that MA, but since SNMP SMI does not allow to
        state in a MIB that an object in a table is an array, the
        information has to be stored in another table with two
        indices, being the first index, the index of the table that
        contains the list or array.

        For all bridges in which the same MAID {dot1agCfmMdFormat,
        dot1agCfmMdName, dot1agCfmMaNetFormat, and dot1agCfmMaNetName}
        is configured, the same set of dot1agCfmMaMepListIdentifiers
        must be configured in the bridges' dot1agCfmMaMepListTables.
        This allows each MEP to determine whether or not it is
        receiving CCMs from all of the other MEPs in the MA.

        For example, if one were creating a new MA whose MAID were
        {charString, 'Dom1', charString, 'MA1'}, that had 2 MEPs, whose
        MEPIDs were 1 and 3, one could, in Bridge A:
         1. Get a new MD index d from dot1agCfmMdTableNextIndex.
         2. Create the Maintenance Domain {charString, 'Dom1'}.
         3. Get a new MA index a from dot1agCfmMdMaNextIndex [d].
         4. Create the Maintenance Association {charString, 'MA1'}.
         5. Create a new dot1agCfmMaMepListEntry for each of the MEPs
            in the MA: [d, a, 1] and [d, a, 3].
         6. Create one of the new MEPs, say [d, a, 1].
        Then, in Bridge B:
         7. Do all of these steps 1-6, except for using the other MEPID
            for the new MEP in Step 6, in this example, MEPID 3.
        Note that, when creating the MA, MEP List Table, and MEP
```

```
         entries in the second bridge, the indices 'd' and 'a'
         identifying the MAID {charString, 'Dom1', charString, 'MA1'}
         may have different values than those in the first Bridge.
        "
    REFERENCE
       "802.1ag clause 12.14.6.1.3:g"
    ::= { dot1agCfmMa 3 }

dot1agCfmMaMepListEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmMaMepListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The known MEPS table entry."
    INDEX { dot1agCfmMdIndex,
            dot1agCfmMaIndex,
            dot1agCfmMaMepListIdentifier
          }
    ::= { dot1agCfmMaMepListTable 1 }

Dot1agCfmMaMepListEntry ::= SEQUENCE {
      dot1agCfmMaMepListIdentifier  Dot1agCfmMepId,
      dot1agCfmMaMepListRowStatus   RowStatus
    }

dot1agCfmMaMepListIdentifier OBJECT-TYPE
    SYNTAX      Dot1agCfmMepId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "MEPID"
    REFERENCE
       "802.1ag clause 12.14.6.1.3:g"
    ::= { dot1agCfmMaMepListEntry 1 }

dot1agCfmMaMepListRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The status of the row. Read SNMPv2-TC (RFC1903) for an
        explanation of the possible values this object can take.
       "
    ::= { dot1agCfmMaMepListEntry 2 }

-- *****************************************************************
-- The MEP Object.  This object represents a Maintenance End
-- Point as described in 802.1ag document.
-- *****************************************************************

-- *****************************************************************
-- The MEP Table
-- *****************************************************************
```

```
dot1agCfmMepTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmMepEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Maintenance Association End Point (MEP) table.

        Each row in the table represents a different MEP.  A MEP is
        an actively managed CFM entity, associated with a specific
        DSAP of a Service Instance, which can generate and receive
        CFM PDUs and track any responses.  It is an end point of a
        single Maintenance Association, and is an endpoint of a
        separate Maintenance Entity for each of the other MEPs in
        the same Maintenance Association (802.1ag clause 3.18).

        This table uses three indices. The first two indices are the
        indices of the Maintenance Domain and MA tables, the reason
        being that a MEP is always related to an MA and Maintenance
        Domain.

        The MEP table also stores all the managed objects for sending
        LBM and LTM.

        *LBM Managed objects

        LBM Managed objects in the MEP table
        enables the management entity to initiate
        transmission of Loopback messages.  It will signal the MEP
        that it should transmit some number of Loopback messages
        and detect the detection (or lack thereof) of the
        corresponding Loopback messages.

        Steps to use entries in this table:

        1) Wait for dot1agCfmMepTransmitLbmStatus value to be
           true.  To do this do this sequence:
           a. an SNMP GET for both SnmpSetSerialNo and
              dot1agCfmMepTransmitLbmStatus objects (in same SNMP
              PDU).
           b. Check if value for dot1agCfmMepTransmitLbmStatus is true.
              - if not, wait x seconds, go to step a above.
              - if yes, save the value of SnmpSetSerialNo and go
                to step 2) below
        2) Change dot1agCfmMepTransmitLbmStatus value from true to
           false to ensure no other management entity will use
           the service. In order to not disturb a possible other NMS
           do this by sending an SNMP SET for both SnmpSetSerialNo
           and dot1agCfmMepTransmitLbmStatus objects (in same SNMP
           PDU,  and make sure SNmpSetSerialNo is the first varBind).
           For the SnmpSetSerialNo varBind, use the value that you
           obtained in step 1)a.. This ensures that two cooperating
           NMSes will not step on each others toes.
        3) Setup the different data to be sent (number of messages,
           optional TLVs,...), except do not set
           dot1agCfmMepTransmitLbmMessages.
        4) Record the current values of dot1agCfmMepLbrIn,
           dot1agCfmMepLbrInOutOfOrder, and dot1agCfmMepLbrBadMsdu.
        6) Set dot1agCfmMepTransmitLbmMessages to a non-zero value to
           initiate transmission of Loopback messages.
        7) Check the value of dot1agCfmMepTransmitLbmResultOK to
           find out if the operation was successfully initiated or
           not.
        8) Monitor the value of dot1agCfmMepTransmitLbmMessages.
           When it reaches 0, the last LBM has been transmitted.
           Wait an additional 5 seconds to ensure that all LBRs have
           been returned.
        9) Compare dot1agCfmMepLbrIn, dot1agCfmMepLbrInOutOfOrder,
           and dot1agCfmMepLbrBadMsdu to their old values from step
           4, above, to get the results of the test.
       10) Change the dot1agCfmMepTransmitLbmStatus value back to
           true to allow other management entities to use the table.

        *LTM Managed objects
        The LTM Managed objects in the MEP table are used in a manner
        similar to that described for LBM transmission, above.  Upon
        successfully initiating the transmission, the variables
        dot1agCfmMepTransmitLtmSeqNumber and
        dot1agCfmMepTransmitLtmEgressIdentifier return the information
        required to recover the results of the LTM from the
        dot1agCfmLtrTable.
       "
    REFERENCE
       "802.1ag clauses 12.14.7 and 19.2"
    ::= { dot1agCfmMep 1 }

dot1agCfmMepEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmMepEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The MEP table entry"
    INDEX { dot1agCfmMdIndex,
            dot1agCfmMaIndex,
            dot1agCfmMepIdentifier
          }
    ::= { dot1agCfmMepTable 1 }

Dot1agCfmMepEntry ::= SEQUENCE {
      dot1agCfmMepIdentifier                  Dot1agCfmMepId,
      dot1agCfmMepIfIndex                     InterfaceIndexOrZero,
      dot1agCfmMepDirection                   Dot1agCfmMpDirection,
      dot1agCfmMepPrimaryVid                  Unsigned32,
      dot1agCfmMepActive                      TruthValue,
      dot1agCfmMepFngState                    Dot1agCfmFngState,
      dot1agCfmMepCciEnabled                  TruthValue,
      dot1agCfmMepCcmLtmPriority              Unsigned32,
      dot1agCfmMepMacAddress                  MacAddress,
      dot1agCfmMepLowPrDef                Dot1agCfmLowestAlarmPri,
      dot1agCfmMepFngAlarmTime                  TimeInterval,
      dot1agCfmMepFngResetTime                  TimeInterval,
      dot1agCfmMepHighestPrDefect           Dot1agCfmHighestDefectPri,
      dot1agCfmMepDefects                       Dot1agCfmMepDefects,
      dot1agCfmMepErrorCcmLastFailure           OCTET STRING,
      dot1agCfmMepXconCcmLastFailure            OCTET STRING,
      dot1agCfmMepCcmSequenceErrors             Counter32,
      dot1agCfmMepCciSentCcms                   Counter32,
      dot1agCfmMepNextLbmTransId                Unsigned32,
      dot1agCfmMepLbrIn                         Counter32,
      dot1agCfmMepLbrInOutOfOrder               Counter32,
      dot1agCfmMepLbrBadMsdu                    Counter32,
      dot1agCfmMepLtmNextSeqNumber              Unsigned32,
      dot1agCfmMepUnexpLtrIn                    Counter32,
      dot1agCfmMepLbrOut                        Counter32,
      dot1agCfmMepTransmitLbmStatus             TruthValue,
      dot1agCfmMepTransmitLbmDestMacAddress     MacAddress,
      dot1agCfmMepTransmitLbmDestMepId          Dot1agCfmMepIdOrZero,
      dot1agCfmMepTransmitLbmDestIsMepId        TruthValue,
      dot1agCfmMepTransmitLbmMessages           Integer32,
      dot1agCfmMepTransmitLbmDataTlv            OCTET STRING,
      dot1agCfmMepTransmitLbmVlanPriority       Integer32,
      dot1agCfmMepTransmitLbmVlanDropEnable     TruthValue,
      dot1agCfmMepTransmitLbmResultOK           TruthValue,
      dot1agCfmMepTransmitLbmSeqNumber          Unsigned32,
      dot1agCfmMepTransmitLtmStatus             TruthValue,
      dot1agCfmMepTransmitLtmFlags              BITS,
      dot1agCfmMepTransmitLtmTargetMacAddress   MacAddress,
      dot1agCfmMepTransmitLtmTargetMepId        Dot1agCfmMepIdOrZero,
      dot1agCfmMepTransmitLtmTargetIsMepId      TruthValue,
      dot1agCfmMepTransmitLtmTtl                Unsigned32,
      dot1agCfmMepTransmitLtmResult             TruthValue,
      dot1agCfmMepTransmitLtmSeqNumber          Unsigned32,
      dot1agCfmMepTransmitLtmEgressIdentifier   OCTET STRING,
      dot1agCfmMepRowStatus                     RowStatus
    }

dot1agCfmMepIdentifier OBJECT-TYPE
    SYNTAX      Dot1agCfmMepId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Integer that is unique among all the MEPs in the same MA.
        Other definition is: a small integer, unique over a given
        Maintenance Association, identifying a specific Maintenance
        association End Point (3.19).

        MEP Identifier is also known as the MEPID.
       "
    REFERENCE
       "802.1ag clauses 3.19, 19.2 and 12.14.7"
    ::= { dot1agCfmMepEntry 1 }

dot1agCfmMepIfIndex OBJECT-TYPE
```

```
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "This object is the interface index of the interface either a
        bridge port, or an aggregated IEEE 802.1 link within a bridge
        port, to which the MEP is attached.

        Upon a restart of the system, the system SHALL, if necessary,
        change the value of this variable so that it indexes the
        entry in the interface table with the same value of ifAlias
        that it indexed before the system restart.  If no such
        entry exists, then the system SHALL set this variable to 0.
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:b"
    ::= { dot1agCfmMepEntry 2 }

dot1agCfmMepDirection OBJECT-TYPE
    SYNTAX      Dot1agCfmMpDirection
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The direction in which the MEP faces on the Bridge port."
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:c and 19.2"
    ::= { dot1agCfmMepEntry 3 }

dot1agCfmMepPrimaryVid OBJECT-TYPE
    SYNTAX      Unsigned32(0..16777215)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "An integer indicating the Primary VID of the MEP, always
        one of the VIDs assigned to the MEP's MA.  The value 0
        indicates that either the Primary VID is that of the
        MEP's MA, or that the MEP's MA is associated with no VID."
   REFERENCE
       "802.1ag clauses 12.14.7.1.3:d"
   DEFVAL { 0 }
    ::= { dot1agCfmMepEntry 4 }

dot1agCfmMepActive OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Administrative state of the MEP

        A Boolean indicating the administrative state of the MEP.

        True indicates that the MEP is to function normally, and
        false that it is to cease functioning."
   REFERENCE
```

```
       "802.1ag clauses 12.14.7.1.3:e and 20.9.1"
   DEFVAL { false }
   ::= { dot1agCfmMepEntry 5 }

dot1agCfmMepFngState OBJECT-TYPE
    SYNTAX      Dot1agCfmFngState
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Current state of the MEP Fault Notification Generator
        State Machine.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:f and 20.35"
    DEFVAL { fngReset }
    ::= { dot1agCfmMepEntry 6 }

dot1agCfmMepCciEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "If set to true, the MEP will generate CCM messages."
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:g and 20.10.1"
    DEFVAL { false }
    ::= { dot1agCfmMepEntry 7 }

dot1agCfmMepCcmLtmPriority OBJECT-TYPE
    SYNTAX      Unsigned32 (0..7)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The priority value for CCMs and LTMs transmitted by the MEP.
        Default Value is the highest priority value allowed to pass
        through the bridge port for any of this MEPs VIDs.
        The management entity can obtain the default value for this
        variable from the priority regeneration table by extracting the
        highest priority value in this table on this MEPs bridge port.
        (1 is lowest, then 2, then 0, then 3-7).
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:h"
    ::= { dot1agCfmMepEntry 8 }

dot1agCfmMepMacAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "MAC address of the MEP."
    REFERENCE
      "802.1ag clause 12.14.7.1.3:i and 19.4"
    ::= { dot1agCfmMepEntry 9 }
```

```
dot1agCfmMepLowPrDef OBJECT-TYPE
    SYNTAX      Dot1agCfmLowestAlarmPri
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "An integer value specifying the lowest priority defect
        that is allowed to generate fault alarm.
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:k and 20.9.5 and Table 20-1"
    DEFVAL { macRemErrXcon }
    ::= { dot1agCfmMepEntry 10}

dot1agCfmMepFngAlarmTime OBJECT-TYPE
    SYNTAX      TimeInterval (250..1000)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The time that defects must be present before a Fault Alarm is
        issued (fngAlarmTime. 20.33.3) (default 2.5s).
       "
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:l and 20.33.3"
    DEFVAL { 250 }
    ::= { dot1agCfmMepEntry 11 }

dot1agCfmMepFngResetTime OBJECT-TYPE
    SYNTAX      TimeInterval (250..1000)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The time that defects must be absent before resetting a
        Fault Alarm (fngResetTime, 20.33.4) (default 10s).
       "
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:m and 20.33.4"
    DEFVAL { 1000 }
    ::= { dot1agCfmMepEntry 12 }

dot1agCfmMepHighestPrDefect OBJECT-TYPE
    SYNTAX   Dot1agCfmHighestDefectPri
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The highest priority defect that has been present since the
        MEPs Fault Notification Generator State Machine was last in
        the FNG_RESET state.
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:n  20.33.9 and Table 21-1"
    ::= { dot1agCfmMepEntry 13 }

dot1agCfmMepDefects OBJECT-TYPE
```

```
    SYNTAX      Dot1agCfmMepDefects
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "A vector of Boolean error conditions from Table 20-1, any of
        which may be true:

        DefRDICCM(0)
        DefMACstatus(1)
        DefRemoteCCM(2)
        DefErrorCCM(3)
        DefXconCCM(4)
       "
    REFERENCE
       ".1ag clauses 12.14.7.1.3:o, 12.14.7.1.3:p, 12.14.7.1.3:q,
        12.14.7.1.3:r, 12.14.7.1.3:s, 20.21.3, 20.23.3, 20.33.5,
        20.33.6, 20.33.7."
    ::= { dot1agCfmMepEntry 14 }

dot1agCfmMepErrorCcmLastFailure OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(1..1522))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The last-received CCM that triggered an DefErrorCCM fault."
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:t and 20.21.2"
    ::= { dot1agCfmMepEntry 15 }

dot1agCfmMepXconCcmLastFailure OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(1..1522))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The last-received CCM that triggered a DefXconCCM fault."
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:u and 20.23.2"
    ::= { dot1agCfmMepEntry 16 }

dot1agCfmMepCcmSequenceErrors OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The total number of out-of-sequence CCMs received from all
        remote MEPs.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:v and 20.16.12"
    ::= { dot1agCfmMepEntry 17 }

dot1agCfmMepCciSentCcms OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Total number of Continuity Check messages transmitted."
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:w and 20.10.2"
    ::= { dot1agCfmMepEntry 18 }

dot1agCfmMepNextLbmTransId OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Next sequence number/transaction identifier to be sent in a
        Loopback message. This sequence number can be zero because
        it wraps around.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.1.3:x and 20.28.2"
    ::= { dot1agCfmMepEntry 19 }

dot1agCfmMepLbrIn OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Total number of valid, in-order Loopback Replies received."
    REFERENCE
       "802.1ag clause 12.14.7.1.3:y and 20.31.1"
    ::= { dot1agCfmMepEntry 20 }

dot1agCfmMepLbrInOutOfOrder OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The total number of valid, out-of-order Loopback Replies
        received.
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:z and 20.31.1"
    ::= { dot1agCfmMepEntry 21 }

dot1agCfmMepLbrBadMsdu OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The total number of LBRs received whose
        mac_service_data_unit did not match (except for the OpCode)
        that of the corresponding LBM (20.2.3).
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:aa  20.2.3"
    ::= { dot1agCfmMepEntry 22}
```

```
dot1agCfmMepLtmNextSeqNumber OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Next transaction identifier/sequence number to be sent in a
        Linktrace message. This sequence number can be zero because
        it wraps around.
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:ab and 20.36.1"
    ::= { dot1agCfmMepEntry 23 }

dot1agCfmMepUnexpLtrIn OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The total number of unexpected LTRs received (20.39.1).
       "
    REFERENCE
       "802.1ag clause 12.14.7.1.3:ac  20.39.1"
    ::= { dot1agCfmMepEntry 24 }

dot1agCfmMepLbrOut OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Total number of Loopback Replies transmitted."
    REFERENCE
       "802.1ag clause 12.14.7.1.3:ad and 20.26.2"
    ::= { dot1agCfmMepEntry 25 }

dot1agCfmMepTransmitLbmStatus OBJECT-TYPE
   SYNTAX      TruthValue
   MAX-ACCESS  read-create
   STATUS      current
   DESCRIPTION
      "A Boolean flag set to true by the bridge port to indicate
       that another LBM may be transmitted.
       Reset to false by the MEP Loopback Initiator State Machine."
   DEFVAL { true }
   ::= { dot1agCfmMepEntry 26 }

dot1agCfmMepTransmitLbmDestMacAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Target MAC Address Field to be transmitted: A unicast
        destination MAC address.
        This address will be used if the value of the column
        dot1agCfmMepTransmitLbmDestIsMepId is 'false'.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.2:b"
    ::= { dot1agCfmMepEntry 27 }

dot1agCfmMepTransmitLbmDestMepId OBJECT-TYPE
    SYNTAX      Dot1agCfmMepIdOrZero
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Maintenance association End Point Identifier of another
        MEP in the same Maintenance Association to which the LBM is
        to be sent.
        This address will be used if the value of the column
        dot1agCfmMepTransmitLbmDestIsMepId is 'true'.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.2:b"
    ::= { dot1agCfmMepEntry 28 }

dot1agCfmMepTransmitLbmDestIsMepId OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "True indicates that MEPID of the target MEP is used for
        Loopback transmission.
        False indicates that unicast destination MAC address of the
        target MEP is used for Loopback transmission.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.2:b"
    ::= {dot1agCfmMepEntry 29 }

dot1agCfmMepTransmitLbmMessages OBJECT-TYPE
    SYNTAX      Integer32(1..1024)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The number of Loopback messages to be transmitted."
    REFERENCE
       "802.1ag clause 12.14.7.3.2:c"
    DEFVAL { 1 }
    ::= {dot1agCfmMepEntry 30 }

dot1agCfmMepTransmitLbmDataTlv OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0..1500))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "An arbitrary amount of data to be included in the Data TLV,
        if the Data TLV is selected to be sent.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.2:d"
    ::= { dot1agCfmMepEntry 31 }

dot1agCfmMepTransmitLbmVlanPriority OBJECT-TYPE
    SYNTAX      Integer32(0..7)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Priority. 3 bit value to be used in the VLAN tag, if present
        in the transmitted frame.

        The default value is CCM priority.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.2:e"
    ::= { dot1agCfmMepEntry 32 }

dot1agCfmMepTransmitLbmVlanDropEnable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Drop Enable bit value to be used in the VLAN tag, if present
        in the transmitted frame.

        For more information about VLAN Drop Enable, please check
        IEEE 802.1ad.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.2:e"
    DEFVAL { true }
    ::= { dot1agCfmMepEntry 33 }

dot1agCfmMepTransmitLbmResultOK OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Indicates the result of the operation:

        - true       The Loopback Message(s) will be
                     (or has been) sent.
        - false      The Loopback Message(s) will not
                     be sent.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.3:a"
    DEFVAL { true }
    ::= { dot1agCfmMepEntry 34 }

dot1agCfmMepTransmitLbmSeqNumber OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Loopback Transaction Identifier
       (dot1agCfmMepNextLbmTransId) of the first LBM (to be) sent.
        The value returned is undefined if
        dot1agCfmMepTransmitLbmResultOK is false.
       "
    REFERENCE
       "802.1ag clause 12.14.7.3.3:a"
    ::= { dot1agCfmMepEntry 35 }

dot1agCfmMepTransmitLtmStatus OBJECT-TYPE
   SYNTAX      TruthValue
   MAX-ACCESS  read-only
   STATUS      current
   DESCRIPTION
      "A Boolean flag set to true by the bridge port to indicate
       that another LTM may be transmitted.
       Reset to false by the MEP Linktrace Initiator State Machine."
   DEFVAL { true }
   ::= { dot1agCfmMepEntry 36 }

dot1agCfmMepTransmitLtmFlags OBJECT-TYPE
    SYNTAX      BITS {
                  useFDBonly   (0)
                }
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The flags field for LTMs transmitted by the MEP."
    REFERENCE
       "802.1ag clause 12.14.7.4.2:b and 20.37.1"
    DEFVAL { {useFDBonly } }
    ::= { dot1agCfmMepEntry 37 }

dot1agCfmMepTransmitLtmTargetMacAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The Target MAC Address Field to be transmitted: A unicast
        destination MAC address.
        This address will be used if the value of the column
        dot1agCfmMepTransmitLtmTargetIsMepId is 'false'.
       "
    REFERENCE
       "802.1ag clause 12.14.7.4.2:c"
    ::= { dot1agCfmMepEntry 38 }

dot1agCfmMepTransmitLtmTargetMepId OBJECT-TYPE
    SYNTAX      Dot1agCfmMepIdOrZero
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
```

```
        "An indication of the Target MAC Address Field to be
         transmitted:
         The Maintenance association End Point Identifier of
         another MEP in the same Maintenance Association
         This address will be used if the value of the column
         dot1agCfmMepTransmitLtmTargetIsMepId is 'true'.
        "
    REFERENCE
        "802.1ag clause 12.14.7.4.2:c"
    ::= { dot1agCfmMepEntry 39 }

dot1agCfmMepTransmitLtmTargetIsMepId OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "True indicates that MEPID of the target MEP is used for
         Linktrace transmission.
         False indicates that unicast destination MAC address of the
         target MEP is used for Loopback transmission.
        "
    REFERENCE
        "802.1ag clause 12.14.7.4.2:c"
    ::= { dot1agCfmMepEntry 40 }

dot1agCfmMepTransmitLtmTtl OBJECT-TYPE
    SYNTAX      Unsigned32 (0..255)
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The LTM TTL field. Default value, if not specified, is 64.
         The TTL field indicates the number of hops remaining to the
         LTM.  Decremented by 1 by each Linktrace Responder that
         handles the LTM.  The value returned in the LTR is one less
         than that received in the LTM.  If the LTM TTL is 0 or 1, the
         LTM is not forwarded to the next hop, and if 0, no LTR is
         generated.
        "
    REFERENCE
        "802.1ag clause 12.14.7.4.2:d and 21.8.4"
    DEFVAL {64}
    ::= { dot1agCfmMepEntry 41 }

dot1agCfmMepTransmitLtmResult OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Indicates the result of the operation:

         - true    The Linktrace Message will be (or has been) sent.
         - false   The Linktrace Message will not be sent"
    REFERENCE
        "802.1ag clause 12.14.7.4.3:a"
    DEFVAL { true }
    ::= { dot1agCfmMepEntry 42 }

dot1agCfmMepTransmitLtmSeqNumber OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The LTM Transaction Identifier
        (dot1agCfmMepLtmNextSeqNumber) of the LTM sent.
        The value returned is undefined if
        dot1agCfmMepTransmitLtmResult is false.
       "
    REFERENCE
       "802.1ag clause 12.14.7.4.3:a"
    ::= { dot1agCfmMepEntry 43 }

dot1agCfmMepTransmitLtmEgressIdentifier OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(8))
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "Identifies the MEP Linktrace Initiator that is originating,
        or the Linktrace Responder that is forwarding, this LTM.
        The low-order six octets contain a 48-bit IEEE MAC address
        unique to the system in which the MEP Linktrace Initiator
        or Linktrace Responder resides.  The high-order two octets
        contain a value sufficient to uniquely identify the MEP
        Linktrace Initiator or Linktrace Responder within that system.

        For most Bridges, the address of any MAC attached to the
        Bridge will suffice for the low-order six octets, and 0 for
        the high-order octets.  In some situations, e.g., if multiple
        virtual Bridges utilizing emulated LANs are implemented in a
        single physical system, the high-order two octets can be used
        to differentiate among the transmitting entities.

        The value returned is undefined if
        dot1agCfmMepTransmitLtmResult is false.
       "
    REFERENCE
       "802.1ag clause 12.14.7.4.3:b and 21.8.8"
    ::= { dot1agCfmMepEntry 44 }

dot1agCfmMepRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
       "The status of the row.

        The writable columns in a row can not be changed if the row
        is active. All columns must have a valid value before a row
        can be activated.
       "
   ::= { dot1agCfmMepEntry 45 }

-- ******************************************************************
-- The Linktrace Reply Table
-- ******************************************************************

dot1agCfmLtrTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmLtrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "This table extends the MEP table and contains a list of
        Linktrace replies received by a specific MEP in response to
        a linktrace message.

        SNMP SMI does not allow to state in a MIB that an object in
        a table is an array.  The solution is to take the index (or
        indices) of the first table and add one or more indices.
       "
    REFERENCE
       "802.1ag clause 12.14.7.5"
    ::= { dot1agCfmMep 2 }

dot1agCfmLtrEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmLtrEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The Linktrace Reply table entry."
    INDEX { dot1agCfmMdIndex,
            dot1agCfmMaIndex,
            dot1agCfmMepIdentifier,
            dot1agCfmLtrSeqNumber,
            dot1agCfmLtrReceiveOrder
          }
    ::= { dot1agCfmLtrTable 1 }

Dot1agCfmLtrEntry ::= SEQUENCE {
      dot1agCfmLtrSeqNumber            Unsigned32,
      dot1agCfmLtrReceiveOrder         Unsigned32,
      dot1agCfmLtrTtl                  Unsigned32,
      dot1agCfmLtrForwarded            TruthValue,
      dot1agCfmLtrTerminalMep          TruthValue,
      dot1agCfmLtrLastEgressIdentifier OCTET STRING,
      dot1agCfmLtrNextEgressIdentifier OCTET STRING,
      dot1agCfmLtrRelay            Dot1agCfmRelayActionFieldValue,
      dot1agCfmLtrChassisIdSubtype     LldpChassisIdSubtype,
      dot1agCfmLtrChassisId            LldpChassisId,
      dot1agCfmLtrManAddressDomain     TDomain,
      dot1agCfmLtrManAddress           TAddress,
      dot1agCfmLtrIngress          Dot1agCfmIngressActionFieldValue,
      dot1agCfmLtrIngressMac           MacAddress,
      dot1agCfmLtrIngressPortIdSubtype LldpPortIdSubtype,
      dot1agCfmLtrIngressPortId        LldpPortId,
      dot1agCfmLtrEgress           Dot1agCfmEgressActionFieldValue,
      dot1agCfmLtrEgressMac            MacAddress,
      dot1agCfmLtrEgressPortIdSubtype  LldpPortIdSubtype,
      dot1agCfmLtrEgressPortId         LldpPortId,
      dot1agCfmLtrOrganizationSpecificTlv  OCTET STRING
    }

dot1agCfmLtrSeqNumber OBJECT-TYPE
    SYNTAX      Unsigned32 (0..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Transaction identifier/Sequence number returned by a previous
        transmit linktrace message command, indicating which LTM's
        response is going to be returned.
       "
    REFERENCE
       "802.1ag clause 12.14.7.5.2:b"
    ::= { dot1agCfmLtrEntry 1}

dot1agCfmLtrReceiveOrder OBJECT-TYPE
    SYNTAX      Unsigned32(1..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "An index to distinguish among multiple LTRs with the same LTR
        Transaction Identifier field value.  dot1agCfmLtrReceiveOrder
        are assigned sequentially from 1, in the order that the
        Linktrace Initiator received the LTRs.
       "
    REFERENCE
       "802.1ag clause 12.14.7.5.2:c"
    ::= { dot1agCfmLtrEntry 2 }

dot1agCfmLtrTtl OBJECT-TYPE
    SYNTAX      Unsigned32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "TTL field value for a returned LTR."
    REFERENCE
       "802.1ag clause 12.14.7.5 and 20.36.2.2"
    ::= { dot1agCfmLtrEntry 3 }

dot1agCfmLtrForwarded OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Indicates if a LTM was forwarded by the responding MP, as
        returned in the 'FwdYes' flag of the flags field.
       "
    REFERENCE
```

```
       "802.1ag clauses 12.14.7.5.3:c and 20.36.2.1"
    ::= { dot1agCfmLtrEntry 4 }

dot1agCfmLtrTerminalMep OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "A boolean value stating whether the forwarded LTM reached a
        MEP enclosing its MA, as returned in the Terminal MEP flag of
        the Flags field.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:d and 20.36.2.1"
    ::= { dot1agCfmLtrEntry 5 }

dot1agCfmLtrLastEgressIdentifier OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(8))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "An octet field holding the Last Egress Identifier returned
        in the LTR Egress Identifier TLV of the LTR.
        The Last Egress Identifier identifies the MEP Linktrace
        Initiator that originated, or the Linktrace Responder that
        forwarded, the LTM to which this LTR is the response.  This
        is the same value as the Egress Identifier TLV of that LTM.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:e and 20.36.2.3"
    ::= { dot1agCfmLtrEntry 6 }

dot1agCfmLtrNextEgressIdentifier OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(8))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "An octet field holding the Next Egress Identifier returned
        in the LTR Egress Identifier TLV of the LTR.  The Next Egress
        Identifier Identifies the Linktrace Responder that
        transmitted this LTR, and can forward the LTM to the next
        hop.  This is the same value as the Egress Identifier TLV of
        the forwarded LTM, if any. If the FwdYes bit of the Flags
        field is false, the contents of this field are undefined,
        i.e., any value can be transmitted, and the field is ignored
        by the receiver.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:f and 20.36.2.4"
    ::= { dot1agCfmLtrEntry 7 }

dot1agCfmLtrRelay OBJECT-TYPE
    SYNTAX      Dot1agCfmRelayActionFieldValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Value returned in the Relay Action field."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:g and 20.36.2.5"
    ::= { dot1agCfmLtrEntry 8 }

dot1agCfmLtrChassisIdSubtype OBJECT-TYPE
    SYNTAX      LldpChassisIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "This object specifies the format of the Chassis ID returned
        in the Sender ID TLV of the LTR, if any.  This value is
        meaningless if the dot1agCfmLtrChassisId has a length of 0."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:h and 21.5.3.2"
    ::= { dot1agCfmLtrEntry 9 }

dot1agCfmLtrChassisId OBJECT-TYPE
    SYNTAX      LldpChassisId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Chassis ID returned in the Sender ID TLV of the LTR, if
        any. The format of this object is determined by the
        value of the dot1agCfmLtrChassisIdSubtype object.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:i and 21.5.3.3"
    ::= { dot1agCfmLtrEntry 10 }

dot1agCfmLtrManAddressDomain OBJECT-TYPE
    SYNTAX      TDomain
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The TDomain that identifies the type and format of
        the related dot1agCfmMepDbManAddress object, used to access
        the SNMP agent of the system transmitting the LTR.  Received
        in the LTR Sender ID TLV from that system.

        Typical values will be one of (not all inclusive) list:

           snmpUDPDomain          (from SNMPv2-TM, RFC3417)
           snmpIeee802Domain      (from SNMP-IEEE802-TM-MIB, RFC4789)

        The value 'zeroDotZero' (from RFC2578) indicates 'no management
        address was present in the LTR', in which case the related
        object dot1agCfmMepDbManAddress must have a zero-length OCTET
        STRING as a value.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:j, 21.5.3.5, 21.9.6"
    ::= { dot1agCfmLtrEntry 11 }

dot1agCfmLtrManAddress OBJECT-TYPE
    SYNTAX      TAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The TAddress that can be used to access the SNMP
        agent of the system transmitting the CCM, received in the CCM
        Sender ID TLV from that system.

        If the related object dot1agCfmLtrManAddressDomain contains
        the value 'zeroDotZero', this object dot1agCfmLtrManAddress
        must have a zero-length OCTET STRING as a value.
      "
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:j, 21.5.3.7, 21.9.6"
    ::= { dot1agCfmLtrEntry 12 }

dot1agCfmLtrIngress OBJECT-TYPE
    SYNTAX      Dot1agCfmIngressActionFieldValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The value returned in the Ingress Action Field of the LTM.
        The value ingNoTlv(0) indicates that no Reply Ingress TLV was
        returned in the LTM."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:k and 20.36.2.6"
    ::= { dot1agCfmLtrEntry 13 }

dot1agCfmLtrIngressMac OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "MAC address returned in the ingress MAC address field.
        If the dot1agCfmLtrIngress object contains the value
        ingNoTlv(0), then the contents of this object are meaningless."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:l and 20.36.2.7"
    ::= { dot1agCfmLtrEntry 14 }

dot1agCfmLtrIngressPortIdSubtype OBJECT-TYPE
    SYNTAX      LldpPortIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Format of the Ingress Port ID.
        If the dot1agCfmLtrIngress object contains the value
        ingNoTlv(0), then the contents of this object are meaningless."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:m and 20.36.2.8"
   ::= { dot1agCfmLtrEntry 15 }
```

```
dot1agCfmLtrIngressPortId OBJECT-TYPE
    SYNTAX      LldpPortId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Ingress Port ID. The format of this object is determined by
        the value of the dot1agCfmLtrIngressPortIdSubtype object.
        If the dot1agCfmLtrIngress object contains the value
        ingNoTlv(0), then the contents of this object are meaningless."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:n and 20.36.2.9"
    ::= { dot1agCfmLtrEntry 16 }

dot1agCfmLtrEgress OBJECT-TYPE
    SYNTAX      Dot1agCfmEgressActionFieldValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The value returned in the Egress Action Field of the LTM.
        The value egrNoTlv(0) indicates that no Reply Egress TLV was
        returned in the LTM."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:o and 20.36.2.10"
    ::= { dot1agCfmLtrEntry 17 }

dot1agCfmLtrEgressMac OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "MAC address returned in the egress MAC address field.
        If the dot1agCfmLtrEgress object contains the value
        egrNoTlv(0), then the contents of this object are meaningless."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:p and 20.36.2.11"
    ::= { dot1agCfmLtrEntry 18 }

dot1agCfmLtrEgressPortIdSubtype OBJECT-TYPE
    SYNTAX      LldpPortIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Format of the egress Port ID.
        If the dot1agCfmLtrEgress object contains the value
        egrNoTlv(0), then the contents of this object are meaningless."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:q and 20.36.2.12"
    ::= { dot1agCfmLtrEntry 19 }

dot1agCfmLtrEgressPortId OBJECT-TYPE
    SYNTAX      LldpPortId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "Egress Port ID. The format of this object is determined by
        the value of the dot1agCfmLtrEgressPortIdSubtype object.
        If the dot1agCfmLtrEgress object contains the value
        egrNoTlv(0), then the contents of this object are meaningless."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:r and 20.36.2.13"
    ::= { dot1agCfmLtrEntry 20 }

dot1agCfmLtrOrganizationSpecificTlv OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(0|4..1500))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "All Organization specific TLVs returned in the LTR, if
        any.  Includes all octets including and following the TLV
        Length field of each TLV, concatenated together."
    REFERENCE
       "802.1ag clauses 12.14.7.5.3:s, 21.5.2"
    ::= { dot1agCfmLtrEntry 21 }

-- ******************************************************************
-- The MEP Database Table
-- ******************************************************************

dot1agCfmMepDbTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Dot1agCfmMepDbEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The MEP Database. A database, maintained by every MEP, that
        maintains received information about other MEPs in the
        Maintenance Domain.

        The SMI does not allow to state in a MIB that an object in
        a table is an array. The solution is to take the index (or
        indices) of the first table and add one or more indices.
       "
    REFERENCE
       "802.1ag clause 19.2.15"
    ::= { dot1agCfmMep 3 }

dot1agCfmMepDbEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmMepDbEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "The MEP Database table entry."
    INDEX { dot1agCfmMdIndex,
            dot1agCfmMaIndex,
            dot1agCfmMepIdentifier,
            dot1agCfmMepDbRMepIdentifier
          }
    ::= { dot1agCfmMepDbTable 1 }
```

```
Dot1agCfmMepDbEntry ::= SEQUENCE {
      dot1agCfmMepDbRMepIdentifier         Dot1agCfmMepId,
      dot1agCfmMepDbRMepState              Dot1agCfmRemoteMepState,
      dot1agCfmMepDbRMepFailedOkTime       TimeStamp,
      dot1agCfmMepDbMacAddress             MacAddress,
      dot1agCfmMepDbRdi                    TruthValue,
      dot1agCfmMepDbPortStatusTlv          Dot1agCfmPortStatus,
      dot1agCfmMepDbInterfaceStatusTlv     Dot1agCfmInterfaceStatus,
      dot1agCfmMepDbChassisIdSubtype       LldpChassisIdSubtype,
      dot1agCfmMepDbChassisId              LldpChassisId,
      dot1agCfmMepDbManAddressDomain       TDomain,
      dot1agCfmMepDbManAddress             TAddress
    }

dot1agCfmMepDbRMepIdentifier OBJECT-TYPE
    SYNTAX      Dot1agCfmMepId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Maintenance association End Point Identifier of a remote MEP
        whose information from the MEP Database is to be returned.
       "
    REFERENCE
       "802.1ag clause 12.14.7.6.2:b"
    ::= { dot1agCfmMepDbEntry 1 }

dot1agCfmMepDbRMepState OBJECT-TYPE
    SYNTAX      Dot1agCfmRemoteMepState
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The operational state of the remote MEP IFF State machines."
    REFERENCE
       "802.1ag clause 12.14.7.6.3:b and 20.22"
    ::= { dot1agCfmMepDbEntry 2}

dot1agCfmMepDbRMepFailedOkTime OBJECT-TYPE
    SYNTAX      TimeStamp
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The time (SysUpTime) at which the IFF Remote MEP state machine
        last entered either the RMEP_FAILED or RMEP_OK state.
       "
    REFERENCE
       "802.1ag clause 12.14.7.6.3:c"
    ::= { dot1agCfmMepDbEntry 3 }

dot1agCfmMepDbMacAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
```

```
       "The MAC address of the remote MEP."
    REFERENCE
       "802.1ag clause 12.14.7.6.3:d and 20.19.7"
    ::= { dot1agCfmMepDbEntry 4 }

dot1agCfmMepDbRdi OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "State of the RDI bit in the last received CCM (true for
        RDI=1), or false if none has been received.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.6.3:e and 20.19.2"
    ::= { dot1agCfmMepDbEntry 5 }

dot1agCfmMepDbPortStatusTlv OBJECT-TYPE
    SYNTAX      Dot1agCfmPortStatus
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "An enumerated value of the Port status TLV received in the
        last CCM from the remote MEP or the default value
        psNoPortStateTLV indicating either no CCM has been received,
        or that nor port status TLV was received in the last CCM.
       "
    REFERENCE
       "802.1ag clause 12.14.7.6.3:f and 20.19.3"
    DEFVAL { psNoPortStateTLV }
    ::= { dot1agCfmMepDbEntry 6}

dot1agCfmMepDbInterfaceStatusTlv OBJECT-TYPE
    SYNTAX      Dot1agCfmInterfaceStatus
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "An enumerated value of the Interface status TLV received
        in the last CCM from the remote MEP or the default value
        isNoInterfaceStatus TLV indicating either no CCM has been
        received, or that no interface status TLV was received in
        the last CCM.
       "
    REFERENCE
       "802.1ag clause 12.14.7.6.3:g and 20.19.4"
    DEFVAL { isNoInterfaceStatusTLV }
    ::= { dot1agCfmMepDbEntry 7}

dot1agCfmMepDbChassisIdSubtype OBJECT-TYPE
    SYNTAX      LldpChassisIdSubtype
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "This object specifies the format of the Chassis ID received
```

```
         in the last CCM."
    REFERENCE
       "802.1ag clauses 12.14.7.6.3:h and 21.5.3.2"
    ::= { dot1agCfmMepDbEntry 8 }

dot1agCfmMepDbChassisId OBJECT-TYPE
    SYNTAX      LldpChassisId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The Chassis ID. The format of this object is determined by the
        value of the dot1agCfmLtrChassisIdSubtype object.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.6.3:h and 21.5.3.3"
    ::= { dot1agCfmMepDbEntry 9 }

dot1agCfmMepDbManAddressDomain OBJECT-TYPE
    SYNTAX      TDomain
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The TDomain that identifies the type and format of
        the related dot1agCfmMepDbManAddress object, used to access
        the SNMP agent of the system transmitting the CCM.  Received
        in the CCM Sender ID TLV from that system.

        Typical values will be one of (not all inclusive) list:

           snmpUDPDomain          (from SNMPv2-TM, RFC3417)
           snmpIeee802Domain      (from SNMP-IEEE802-TM-MIB, RFC4789)

        The value 'zeroDotZero' (from RFC2578) indicates 'no management
        address was present in the LTR', in which case the related
        object dot1agCfmMepDbManAddress must have a zero-length OCTET
        STRING as a value.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.6.3:h, 21.5.3.5, 21.6.7"
    ::= { dot1agCfmMepDbEntry 10 }

dot1agCfmMepDbManAddress OBJECT-TYPE
    SYNTAX      TAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "The TAddress that can be used to access the SNMP
        agent of the system transmitting the CCM, received in the CCM
        Sender ID TLV from that system.

        If the related object dot1agCfmMepDbManAddressDomain contains
        the value 'zeroDotZero', this object dot1agCfmMepDbManAddress
        must have a zero-length OCTET STRING as a value.
       "
    REFERENCE
       "802.1ag clauses 12.14.7.6.3:h, 21.5.3.7, 21.6.7"
    ::= { dot1agCfmMepDbEntry 11 }

-- ******************************************************************
-- NOTIFICATIONS (TRAPS)
-- These notifications will be sent to the management entity
-- whenever a MEP loses/restores contact with one or more other MEPs.
-- ******************************************************************

dot1agCfmFaultAlarm NOTIFICATION-TYPE
    OBJECTS     { dot1agCfmMepHighestPrDefect
                }
    STATUS      current
    DESCRIPTION
       "A MEP has a persistent defect condition. A notification
        (fault alarm) is sent to the management entity with the OID
        of the MEP that has detected the fault.

        Whenever a MEP has a persistent defect,
        it may or may not generate a Fault Alarm to warn the system
        administrator of the problem, as controlled by the MEP
        Fault Notification Generator State Machine and associated
        Managed Objects. Only the highest-priority defect, as shown
        in Table 20-1, is reported in the Fault Alarm.

        If a defect with a higher priority is raised after a Fault
        Alarm has been issued, another Fault Alarm is issued.

        The management entity receiving the notification can identify
        the system from the network source address of the
        notification, and can identify the MEP reporting the defect
        by the indices in the OID of the dot1agCfmMepHighestPrDefect
        variable in the notification:

           dot1agCfmMdIndex - Also the index of the MEP's
                              Maintenance Domain table entry
                              (dot1agCfmMdTable).
           dot1agCfmMaIndex - Also an index (with the MD table index)
                              of the MEP's Maintenance Association
                              network table entry
                              (dot1agCfmMaNetTable), and (with the MD
                              table index and component ID) of the
                              MEP's MA component table entry
                              (dot1agCfmMaCompTable).
           dot1agCfmMepIdentifier - MEP Identifier and final index
                              into the MEP table (dot1agCfmMepTable).
       "
    REFERENCE
       "802.1ag clause 12.14.7.7"
    ::= { dot1agNotifications 1 }

-- ******************************************************************
-- IEEE 802.1ag MIB Module - Conformance Information
-- ******************************************************************

dot1agCfmCompliances OBJECT IDENTIFIER ::= { dot1agCfmConformance 1 }
dot1agCfmGroups      OBJECT IDENTIFIER ::= { dot1agCfmConformance 2 }

-- ******************************************************************
-- Units of conformance
-- ******************************************************************
dot1agCfmStackGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmStackMdIndex,
      dot1agCfmStackMaIndex,
      dot1agCfmStackMepId,
      dot1agCfmStackMacAddress
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Stack group."
    ::= { dot1agCfmGroups 1 }

dot1agCfmDefaultMdGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmDefaultMdDefLevel,
      dot1agCfmDefaultMdDefMhfCreation,
      dot1agCfmDefaultMdDefIdPermission,
      dot1agCfmDefaultMdStatus,
      dot1agCfmDefaultMdLevel,
      dot1agCfmDefaultMdMhfCreation,
      dot1agCfmDefaultMdIdPermission
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Default MD Level group."
    ::= { dot1agCfmGroups 2 }

dot1agCfmVlanIdGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmVlanPrimaryVid,
      dot1agCfmVlanRowStatus
   }
    STATUS      current
    DESCRIPTION
       "Objects for the VLAN ID group."
    ::= { dot1agCfmGroups 3 }

dot1agCfmConfigErrorListGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmConfigErrorListErrorType
    }
    STATUS current
    DESCRIPTION
       "Objects for the CFM Configuration Error List Group."
    ::= {dot1agCfmGroups 4 }
```

```
dot1agCfmMdGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmMdTableNextIndex,
      dot1agCfmMdName,
      dot1agCfmMdFormat,
      dot1agCfmMdMdLevel,
      dot1agCfmMdMhfCreation,
      dot1agCfmMdMhfIdPermission,
      dot1agCfmMdMaNextIndex,
      dot1agCfmMdRowStatus
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Maintenance Domain Group."
    ::={dot1agCfmGroups 5 }

dot1agCfmMaGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmMaNetFormat,
      dot1agCfmMaNetName,
      dot1agCfmMaNetCcmInterval,
      dot1agCfmMaNetRowStatus,
      dot1agCfmMaCompPrimaryVlanId,
      dot1agCfmMaCompMhfCreation,
      dot1agCfmMaCompIdPermission,
      dot1agCfmMaCompRowStatus,
      dot1agCfmMaCompNumberOfVids,
      dot1agCfmMaMepListRowStatus
    }
    STATUS      current
    DESCRIPTION
       "Objects for the MA group."
    ::= { dot1agCfmGroups 6 }

dot1agCfmMepGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmMepIfIndex,
      dot1agCfmMepDirection,
      dot1agCfmMepPrimaryVid,
      dot1agCfmMepActive,
      dot1agCfmMepFngState,
      dot1agCfmMepCciEnabled,
      dot1agCfmMepCcmLtmPriority,
      dot1agCfmMepMacAddress,
      dot1agCfmMepLowPrDef,
      dot1agCfmMepFngAlarmTime,
      dot1agCfmMepFngResetTime,
      dot1agCfmMepHighestPrDefect,
      dot1agCfmMepDefects,
      dot1agCfmMepErrorCcmLastFailure,
      dot1agCfmMepXconCcmLastFailure,
      dot1agCfmMepCcmSequenceErrors,
      dot1agCfmMepCciSentCcms,
      dot1agCfmMepNextLbmTransId,
      dot1agCfmMepLbrIn,
      dot1agCfmMepLbrInOutOfOrder,
      dot1agCfmMepLbrBadMsdu,
      dot1agCfmMepLtmNextSeqNumber,
      dot1agCfmMepUnexpLtrIn,
      dot1agCfmMepLbrOut,
      dot1agCfmMepTransmitLbmStatus,
      dot1agCfmMepTransmitLbmDestMacAddress,
      dot1agCfmMepTransmitLbmDestMepId,
      dot1agCfmMepTransmitLbmDestIsMepId,
      dot1agCfmMepTransmitLbmMessages,
      dot1agCfmMepTransmitLbmDataTlv,
      dot1agCfmMepTransmitLbmVlanPriority,
      dot1agCfmMepTransmitLbmVlanDropEnable,
      dot1agCfmMepTransmitLbmResultOK,
      dot1agCfmMepTransmitLbmSeqNumber,
      dot1agCfmMepTransmitLtmStatus,
      dot1agCfmMepTransmitLtmFlags,
      dot1agCfmMepTransmitLtmTargetMacAddress,
      dot1agCfmMepTransmitLtmTargetMepId,
      dot1agCfmMepTransmitLtmTargetIsMepId,
      dot1agCfmMepTransmitLtmTtl,
      dot1agCfmMepTransmitLtmResult,
      dot1agCfmMepTransmitLtmSeqNumber,
      dot1agCfmMepTransmitLtmEgressIdentifier,
      dot1agCfmMepRowStatus,
      dot1agCfmLtrForwarded,
      dot1agCfmLtrRelay,
      dot1agCfmLtrChassisIdSubtype,
      dot1agCfmLtrChassisId,
      dot1agCfmLtrManAddress,
      dot1agCfmLtrManAddressDomain,
      dot1agCfmLtrIngress,
      dot1agCfmLtrIngressMac,
      dot1agCfmLtrIngressPortIdSubtype,
      dot1agCfmLtrIngressPortId,
      dot1agCfmLtrEgress,
      dot1agCfmLtrEgressMac,
      dot1agCfmLtrEgressPortIdSubtype,
      dot1agCfmLtrEgressPortId,
      dot1agCfmLtrTerminalMep,
      dot1agCfmLtrLastEgressIdentifier,
      dot1agCfmLtrNextEgressIdentifier,
      dot1agCfmLtrTtl,
      dot1agCfmLtrOrganizationSpecificTlv
    }
    STATUS      current
    DESCRIPTION
       "Objects for the MEP group."
    ::= { dot1agCfmGroups 7 }

dot1agCfmMepDbGroup OBJECT-GROUP
    OBJECTS {
      dot1agCfmMepDbRMepState,
      dot1agCfmMepDbRMepFailedOkTime,
      dot1agCfmMepDbMacAddress,
      dot1agCfmMepDbRdi,
      dot1agCfmMepDbPortStatusTlv,
      dot1agCfmMepDbInterfaceStatusTlv,
      dot1agCfmMepDbChassisIdSubtype,
      dot1agCfmMepDbChassisId,
      dot1agCfmMepDbManAddressDomain,
      dot1agCfmMepDbManAddress
    }
    STATUS      current
    DESCRIPTION
       "Objects for the MEP group."
    ::= { dot1agCfmGroups 8 }

dot1agCfmNotificationsGroup NOTIFICATION-GROUP
    NOTIFICATIONS {
      dot1agCfmFaultAlarm
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Notifications group."
    ::= { dot1agCfmGroups 9 }

-- ******************************************************************
-- MIB Module Compliance statements
-- ******************************************************************

dot1agCfmCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
       "The compliance statement for support of the CFM MIB module."
    MODULE
        MANDATORY-GROUPS {
            dot1agCfmStackGroup,
            dot1agCfmDefaultMdGroup,
            dot1agCfmConfigErrorListGroup,
            dot1agCfmMdGroup,
            dot1agCfmMaGroup,
            dot1agCfmMepGroup,
            dot1agCfmMepDbGroup,
            dot1agCfmNotificationsGroup
         }

    GROUP dot1agCfmVlanIdGroup
    DESCRIPTION "The VLAN ID group is optional."

    OBJECT dot1agCfmMepLbrBadMsdu
    MIN-ACCESS not-accessible
    DESCRIPTION "The dot1agCfmMepLbrBadMsdu variable is optional.  It
                 must not be present if the system cannot compare a
                 received LBR to the corresponding LBM."

    OBJECT dot1agCfmMdRowStatus
```

```
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    OBJECT dot1agCfmMaNetRowStatus
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    OBJECT dot1agCfmMaCompRowStatus
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    OBJECT dot1agCfmVlanRowStatus
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    OBJECT  dot1agCfmMaMepListRowStatus
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    OBJECT  dot1agCfmMepRowStatus
      SYNTAX       RowStatus { active(1), notInService(2) }
      WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                               destroy(6) }
      DESCRIPTION "Support for createAndWait is not required."

    ::= { dot1agCfmCompliances 1 }

END
```

Insert a new Clause 18 as follows.

18. Principles of Connectivity Fault Management operation

Connectivity Fault Management (CFM) comprises capabilities for detecting, verifying, and isolating connectivity failures in Virtual Bridged Local Area Networks. These capabilities can be used in networks operated by multiple independent organizations, each with restricted management access to each other's equipment.

CFM is designed to be transparent to the customer data transported by a network and to be capable of providing maximum fault coverage. Accordingly, CFM Entities (Clause 19) are specified as shims that make use of and provide the ISS or EISS (6.6, 6.15) at Service Access Points (SAPs) within the network. They can, in principle, be added between any of the other media-independent protocol entities that compose a Bridge Port, without requiring changes to those entities. Customer data is forwarded transparently by CFM Entities while CFM PDUs are generated and processed as specified in Clause 20. The formats of the various CFM PDUs are described in Clause 21, and the creation and placement of CFM Entities in Bridges are specified in Clause 22.

This clause provides the context necessary to understand each of the CFM protocols, and how CFM Entities in bridges are selected and configured as Maintenance Points (MPs, 19.1) to participate in and operate those protocols.

CFM introduces the following concepts to support multiple independent operators, each supporting service instances for multiple independent customers:

a) A Maintenance Domain (18.1) is a part of a network that is controlled by a single operator and used to support connectivity between the Domain Service Access Points (DoSAPS, 18.1) that bound the Maintenance Domain.
b) A Maintenance Domain Level (MD Level, 18.3), carried in CFM PDUs, allows each of an operator's customers also to use CFM and to function as an operator if desired, multiplexing provided service instances over its own connectivity.
c) A Maintenance Association (MA, 18.2) is created by configuring CFM Entities that support an individual service instance's DoSAPs as Maintenance association End Points (MEPs), and is used to monitor connectivity provided by that instance through the Maintenance Domain.

Maintenance Point (19.1) configuration of CFM Entities supports the hierarchical nesting of Maintenance Domains. The DoSAPs for a given service instance are Intermediate Service Access Points (ISAPs) for MAs monitoring connectivity through an enclosing Maintenance Domain. When a MEP is configured, a Maintenance association Intermediate Point (MIP) can also be configured at the appropriate level for its immediately enclosing Maintenance Domain. Below the innermost Maintenance Domain, every physical LAN can serve as an implied Maintenance Domain, and every Bridge Port as an implied MEP. In the innermost Maintenance Domain, every Bridge Port is an ISAP and can be configured with a MIP.

Maintenance Domains are nested, not overlapped. That is, if a given DoSAP for service instance 1 in Maintenance Domain *x* is an ISAP for Maintenance Domain *y*, then all DoSAPs for service instance 1 in Maintenance Domain *x* are either ISAPs or DoSAPs for Maintenance Domain *y*, and no other Maintenance Domain. Conformance to this rule is not a condition for the correct operation of CFM. Rather, if this rule is violated, then CFM will detect the violation as an error condition, whatever the intentions of the administrators.

The information about individual service instances that is configured and recorded within a network to support CFM is entirely associated with MEPs, and thus scales linearly with the number of SAPs provided to customers. The transmission of CFM PDUs is stimulated by state machines associated with MEPs as are actions taken that require recording information from received PDUs. MIPs can add, check, and respond to information in received PDUs, thus supporting discovery of paths among MEPs and location of faults along those paths. MIPs can be configured per Maintenance Domain, thus allowing the amount of MIP configuration information to scale linearly with the size of the network, and is thus a constant function of the ability of the network to support those MIPs, rather than being a product of the number of service instances supported. In contrast each MEP is associated with a SAP that provides access to a single service instance.

CFM functions are partitioned as follows:

- Path discovery
- Fault detection
- Fault verification and isolation
- Fault notification
- Fault recovery

Path discovery uses the Linktrace protocol (20.3) to determine the path taken to a target MAC address, MIP by MIP, from one MEP to another MP across an MA. This target MAC address can be that of a MIP, a MEP, or any other Individual MAC address. A Linktrace Message (LTM, 20.3.1) is multicast from a MEP to its neighboring MIPs, and from MIP to MIP, to the MP terminating the path. Each MIP along the path and the terminating MP return unicast Linktrace Replies (LTR, 20.3.2) to the originating MEP. LTMs are triggered by administrative action. The replies are cataloged by the originating MEP for examination by the administrator.

Fault detection uses the Continuity Check protocol (20.1) to detect both connectivity failures and unintended connectivity between service instances. Each MEP can periodically transmit a multicast Connectivity Check Message (CCM) announcing the identity of the MEP and its MA, and tracks the CCMs received from the other MEPs. All connectivity faults that can misdirect a CCM show up as differences between the CCMs received and the MEP’s configured expectations. The state of the tracked CCMs is available to the administrator for examination.

Fault verification and fault isolation are administrative actions, typically performed after fault detection. Fault verification can also confirm successful initiation or restoration of connectivity. The administrator uses the Loopback protocol (20.2) to perform fault verification. A MEP can be ordered to transmit a unicast Loopback Message (LBM, 20.2.1) to a MEP or MIP in the MA, whose MAC address can be discovered from CCMs (MEPs only) or LTMs/LTRs (MEPs or MIPs). The receiving MP responds by transforming the LBM into a unicast Loopback Reply (LBR, 20.2.2) sent back to the originating MEP. That MEP records the responses for examination by the administrator.

Fault notification is provided by a MEP that has detected a connectivity fault in its MA, either because expected CCMs were not received, because unexpected or invalid CCMs were received, or because a CCM carried a notification of the failure of its associated Bridge Port.

Fault recovery is provided by the Spanning Tree Protocols (Clause 13) and by activities by the network administrator, such as the correction of configuration errors or replacement of failed components that are outside the scope of this standard.

18.1 Maintenance Domains and Domain Service Access Points

A “domain” may be defined as a network or part of a network that is capable of offering connectivity to systems outside this region. It comprises a set of DoSAPs at which the domain is capable of or intended to offer connectivity to systems outside the domain. Each DoSAP is an instance either of the EISS or of the ISS. A Maintenance Domain is then a domain or part of a domain for which faults in connectivity are to be managed.

A Maintenance Domain is, or is intended to be, fully connected internally. That is, a DoSAP associated with a Maintenance Domain has connectivity to every other DoSAP in the Maintenance Domain, in the absence of faults. It is, in principle at least, capable of connecting (i.e., supporting service between) any of its DoSAPs. The factors (apart from faults) that can prevent desired connectivity are second-order, e.g., exhaustion of resources that would support a proposed connection, an administrative decision to limit multipoint services, misconfiguration of security parameters, unwillingness of customers or attached Maintenance Domains to use a particular connection configuration. The purpose of CFM is to monitor and diagnose the ability of a Maintenance Domain to meet its operators' intentions for connectivity, as expressed in the configuration of CFM.

Figure 18-1 illustrates a single Maintenance Domain, consisting of five Bridges, from the point of view of an operator. The operator has six DoSAPs to offer to customers.

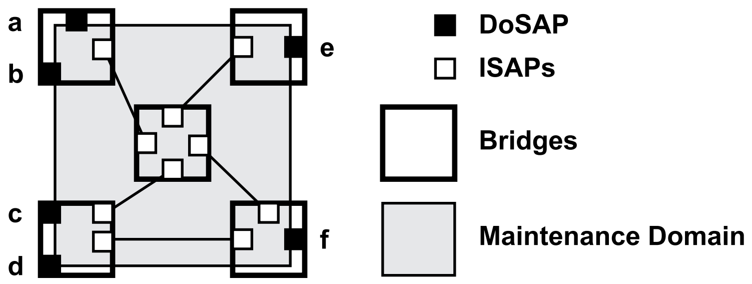

Figure 18-1—One Maintenance Domain: operator's view

Each Maintenance Domain can be separately administered. Each Maintenance Domain is assigned a Maintenance Domain Name. This should be chosen to be unique among all those used or available to an operator, and to facilitate easy identification of administrative responsibility for the Maintenance Domain. CFM PDU formats support the creation of Maintenance Domain Names that are unique over the domain for which CFM is to protect against accidental concatenation of service instances. Ideally, these names are globally unique.

There are a number of methods to fulfill the requirements to both maintain connectivity within, and control access to, a Maintenance Domain. A Maintenance Domain can be, for example:

a) An MST Region, identified and kept separate from other Maintenance Domains, by its MST Configuration Name (8.9.2, 13.7);
b) A number of such MST Regions (13.5), interconnected via the CIST (13.4), and kept separate from other Maintenance Domains by configuring the managed objects controlling the CIST topology; or
c) An entire Virtual Bridged Local Area Network, and kept separate from other Maintenance Domains by means not specified in this standard.
d) A subset of the DoSAPs of a VLAN Bridged Network, assigned to the use of a particular higher-level provider among many.
e) A subset of the DoSAPs of the concatenated network in item c).
f) A single physical link between two providers or between a provider and a customer.
g) A single VLAN on a single physical link between two providers or between a provider and a customer.

18.2 Service instances and Maintenance Associations

When a network administrator configures a service instance for a customer, the network administrator configures some number of the DoSAPs to be accessible by that customer, assigning whatever identifiers the systems require to segregate that service instance from others supported by that Maintenance Domain (e.g., VIDs), as well as configuring other service properties (such as bandwidth profiles) for the DoSAPs. Creation of a service instance establishes a connectionless connectivity (6.1.8) among the selected DoSAPs.

Configuring a Maintenance association End Point (MEP) at each of the DoSAPs of a service instance creates a Maintenance Association (MA) to monitor that connectionless connectivity. The MA is identified by a Short MA Name that is unique within the Maintenance Domain. Together, the Maintenance Domain Name and the Short MA Name form the Maintenance Association Identifier (MAID) that is carried in CFM PDUs to identify inadvertent connections among MEPs. A small integer, the Maintenance association End Point Identifier (MEPID) uniquely identifies each MEP among those configured on a single MA.

NOTE 1—In order to accommodate the requirements of service providers as expressed in ITU-T Y.1731 (2006), the Maintenance Domain Name can be configured to be null, in which case the Short MA Name needs to be globally unique, in order to provide complete protection against the accidental concatenation of service instances. Suitable choices for globally unique Short MA Names are defined in ITU-T Y.1731.

Figure 18-2 illustrates a single service instance created from the Maintenance Domain of Figure 18-1, again from the point of view of an operator. It offers four DoSAPs to a customer (C1). Each DoSAP is marked with its MEPID.

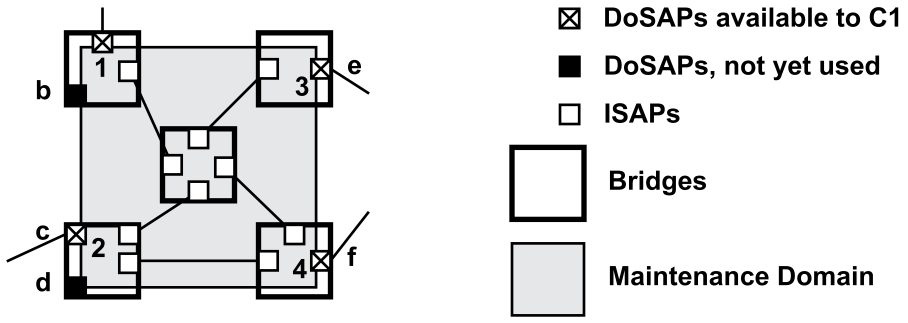

Figure 18-2—One service instance: operator's view

Figure 18-3 illustrates that same service instance from the point of view of customer C1. The customer has four items of customer equipment attached to the four DoSAPs 1–4. The means by which the operator connects the four DoSAPs to create the service instance are invisible to the customer.

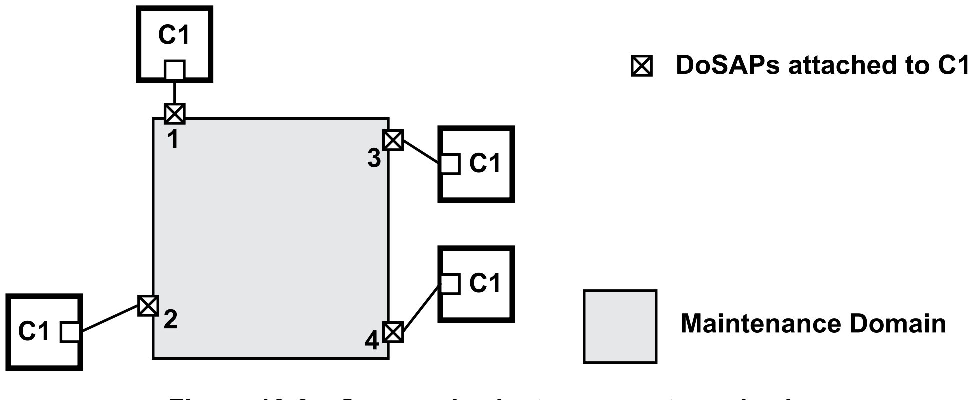

Figure 18-3—One service instance: customer's view

NOTE 2—The term "service instance" has been used to mean an "end-to-end service supplied to a customer." One cannot predict whether an apparently end-to-end service instance will be used by a customer to create another service instance at a still higher level to yet another customer. Therefore, no specific term is provided by this standard to label an "end-to-end" service instance.

The CFM protocol state machines and variables (see Clause 20) are used by MEPs to conduct protocol exchanges within a Maintenance Domain. Each MIP functions symmetrically with respect to the service access points of its containing MA and service instance, and is modeled as comprising two MIP Half Functions (MHFs). Figure 18-4 specifies the symbols used for MEPs, MHFs, and MIPs in this standard, e.g., in Figure 18-7.

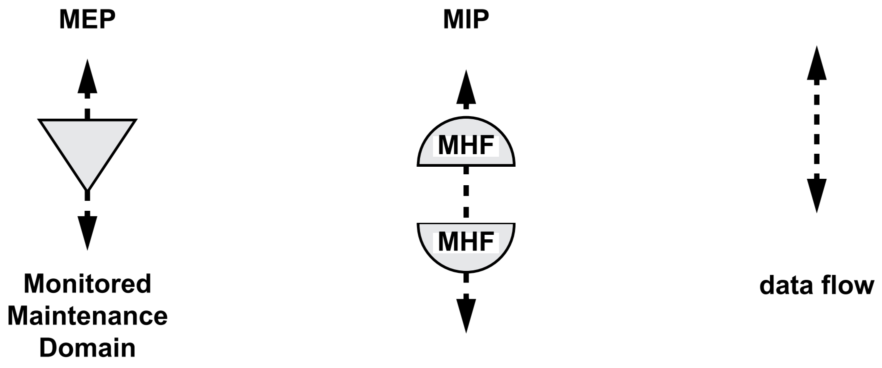

Figure 18-4—MEP and MIP Symbols

18.3 Maintenance Domain Levels

A Maintenance Domain can provide service instances to an enclosing Maintenance Domain or utilize service instances provided by an enclosed Maintenance Domain. An explicit Maintenance Domain Level (MD Level) is included in CFM PDUs to indicate the nesting relationships among Maintenance Domains, because systems can be organized into Maintenance Domains as administrative needs dictate, without necessarily adding headers to data frames to enforce a layered organization.

As part of the nested decomposition facility provided by Maintenance Domains, the CFM protocols hide information. The components of a given Maintenance Domain interior to the DoSAPs presented to the enclosing Maintenance Domain are invisible, through the CFM protocols and managed objects, to the Maintenance Domains enclosing that given Maintenance Domain, except for configured points of visibility. No part of a Maintenance Domain lower than (nested within) a given Maintenance Domain is unintentionally visible to any higher Maintenance Domain. No part of a Maintenance Domain is ever visible to any Maintenance Domain except its immediately higher (enclosing) Maintenance Domain. This enables the separation of responsibility for network administration. The administrator of an end-to-end service on the largest scale can be insulated from the administration of the networks comprising that service, and so on, in principle, down to the administration of individual LANs. This prevents, for example, a customer from learning the internal topology of an operator’s network.

In order to facilitate the diagnosis of connectivity failures, an administrator can make a DoSAP visible as an ISAP to the immediately enclosing Maintenance Domain by configuring it as a Maintenance association Intermediate Point (MIP). The diagnostic functions provided by CFM detect connectivity failures between any pair of MEPs in an MA. The MIPs allow these failures to be isolated to smaller segments of the network; loss of connectivity between a pair of MIPs corresponds to a MEP connectivity failure at a lower MD Level. In the lowest Maintenance Domains, the MIPs can be configured on individual Bridge Ports.

Figure 18-5 illustrates the nesting of Maintenance Domains, service instances, and MAs. It illustrates nested service instances, each configured with one MA in a Maintenance Domain. There are seven Maintenance Domains, and hence seven service instances illustrated. At the (white) “provider” MD Level, the service instance is offered to a single “customer” C1. That customer creates a (gray) service instance of his own, configured with an MA that verifies the integrity of the service instance offered by the provider. Several levels of nested Maintenance Domains are shown:

a) Covering the widest physical extent, a Maintenance Domain (and MA and service instance) at the customer MD Level has a MEP in each of four devices labeled C1 in Figure 18-5. The four DoSAPs a, b, g, and h can be configured as MIPs for this MA.
b) Covering the physical range corresponding to the limits of the end-to-end provider's network are the MEPs a, b, g, and h belonging to the provider MD Level. The four DoSAPs c, d, e, and f can be configured as MIPs for this MA.
c) Each of the two operator networks has a Maintenance Domain at the operator MD Level. These DoSAPs are marked a through h. The unlabeled SAPs in both operators' Maintenance Domains can be configured as MIPs for their respective MAs.
d) Three "physical" layer Maintenance Domains are shown as well. (Two are labeled "PMDL," for "physical MD Level.") Note that not all physical links are given Maintenance Domains.
    1) A network of devices is being used, in some unspecified fashion, to offer the operators' Bridges an emulated shared-medium LAN with four DoSAPs. A Maintenance Domain has been created to verify the connectivity of that LAN.

NOTE—This central emulated LAN could equally be a network of Bridges that are separated via non-standard configuration from the operator clouds on either side, a network of IEEE P802.1ah Bridges, a network of Ethernet over SONET circuits with a central interconnect device, or an ATM Emulated LAN.[9]

    2) Physical level Maintenance Domains have been created at the edge of the network (on the left side) to monitor two LANs, perhaps because they include Repeaters or are otherwise unreliable.

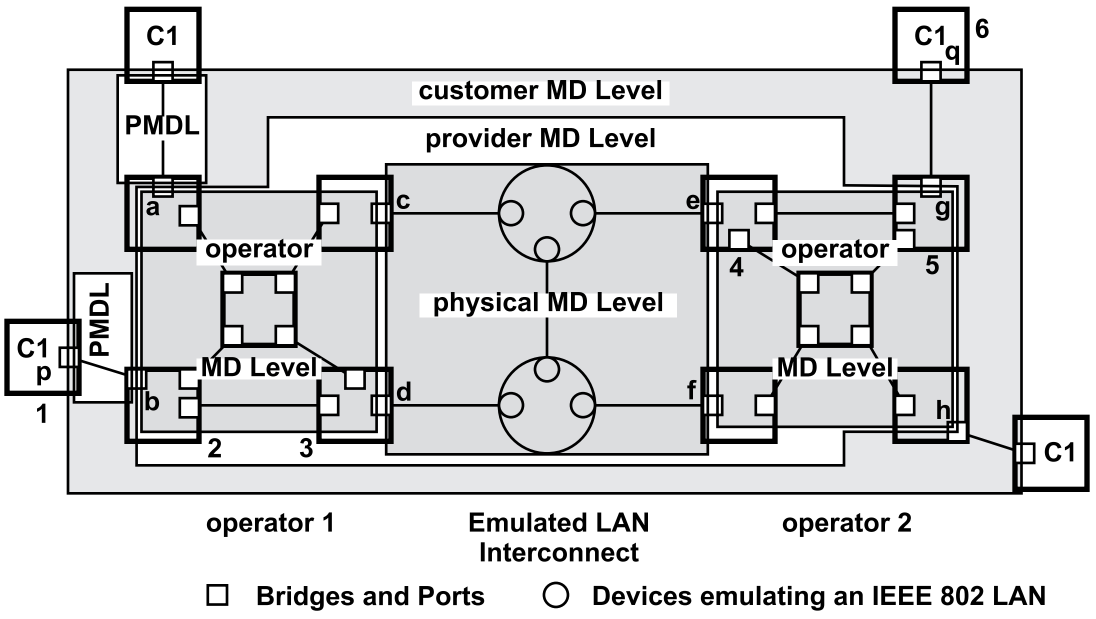

Figure 18-5—Maintenance Associations: one service instance in a provider network

Figure 18-6 expands the upper right corner of Figure 18-5, including Bridges 4 and 5 and device 6. Examining device 6 (C1), we see that the highest MA, at the "customer" MD Level, requires a MEP in its Port q. The side of the DoSAP of this service instance that offers access to the service instance faces the center of device 6, just like the service offered by a LAN. On the other hand, in Port g, there are two service instances, each with a MEP: the white service instance at the provider MD Level, and the gray service instance at the operator MD Level. Neither of these service instances include the wire attached to Port g. Thus, ensuring the proper operation of the LAN connecting Port q to Port g is the responsibility of the MA at the customer MD Level, and not the other MAs. Port e has two MEPs, one facing in each direction.

[9]See ATM Forum af-lane-0021.000; "LAN Emulation over ATM 1.0"; http://www.mfaforum.org/tech/atm_specs.shtml.

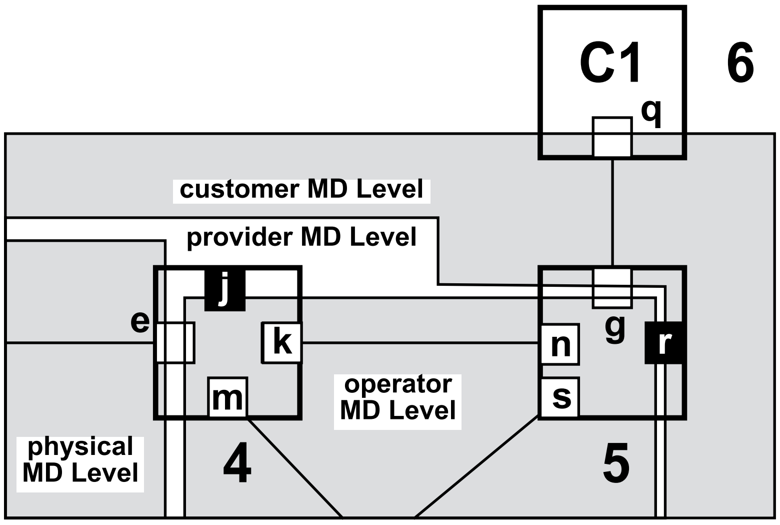

Figure 18-6—Maintenance Associations: Expansion of Figure 18-5

Port g in Figure 18-6 also is an example of the fact that only the MD Level distinguishes the provider and operator Level Maintenance Domains at Port g. The single service instance in the example network shown in Figure 18-5 and Figure 18-6 is carried on a single VLAN with a single VID through the right-hand operator's Maintenance Domain. Within that Maintenance Domain, it cannot be distinguished whether a data frame on that VID belongs to the operator, provider, or customer Maintenance Domain; it belongs to all of them. However, the MD Level permits CFM PDUs to be associated with different Maintenance Domains even when they share the same VID.

Port j and Port r have been added to Figure 18-6; they are not present in Figure 18-5. Both represent DoSAPs that have not been configured to support either a service instance or an MA. For the integrity of the network to be assured by Connectivity Fault Management, these DoSAPs should be configured to be Disabled.

Ports k, m, n, and s can be configured as ISAPs for the gray service instance, providing a MIP at the operator MD Level. Port e can be configured with a MIP for the provider MD Level, and Port g with a MIP for the customer MD Level.

Figure 18-7 illustrates the use of MD Levels to allow the use of CFM by a user of connectivity provided by two bridged Maintenance Domains and by the operators of each of those Maintenance Domains. It illustrates a vertical slice through Figure 18-5, including two of the customer's devices, and the path through the network connecting them. One can see from Figure 18-7 why MD Levels are required to identify CFM PDUs; there would otherwise be no way for Bridge Port b, for example, to tell whether a given CFM PDU is to be sunk (MD Levels 2 or 3) or passed (MD Level 5).

Assignment of numerical MD Levels to the "customer" role, the "service provider" role, or the "operator" MD Levels is somewhat arbitrary, since those terms imply business relationships that cannot be standardized. See J.3 for a more detailed discussion of the assignment of MD Levels.

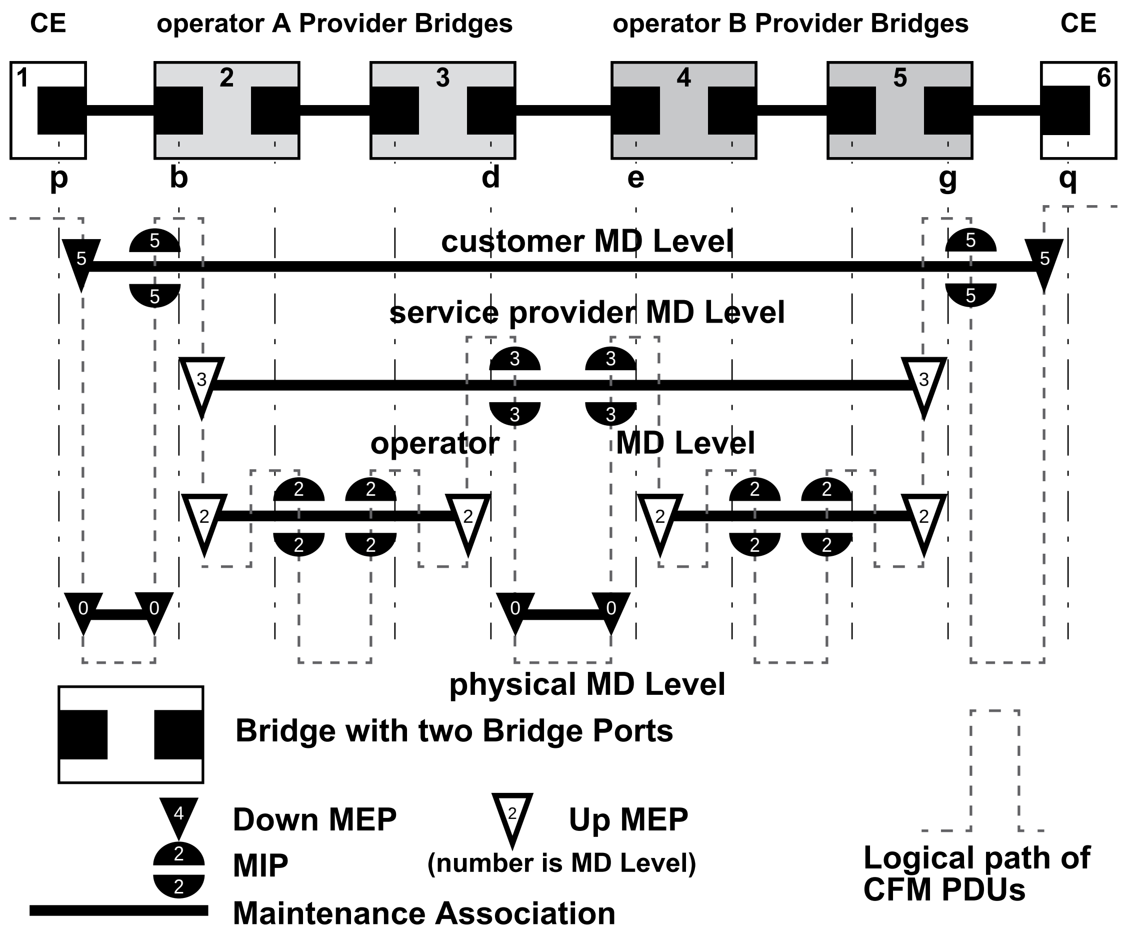

Figure 18-7—MEPs, MIPs, and MD Levels

Insert a new Clause 19 as follows.

19. Connectivity Fault Management Entity operation

This clause specifies:

a) How the Maintenance Points (MPs) that instantiate a CFM Maintenance Association (MA) attach to ISS-SAPs or EISS-SAPs (19.1);
b) How MPs are addressed in networks (19.4); and
c) The architecture of Maintenance association End Points (MEPs, 19.2), Maintenance association Intermediate Points (MIPs, 19.3), MIP Half Functions (MHFs, 19.3), Linktrace Output Multiplexers (LOMs, 19.5), and Linktrace Responders (19.6), including their identification, addressing, and decomposition into components that provide the CFM functions.

NOTE—Clause 18 introduces the principles of CFM operation and the network architectural concepts that support it. Clause 20 specifies the protocols operated by the components of each MP, and Clause 21 specifies the PDU formats used by those protocols. The use of CFM within systems and networks is further described in Clause 22.

The models of operation in this clause provide a basis for specifying the externally observable behavior of MEPs and MIPs, and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified. Conformance of equipment to this standard is purely in respect of observable protocol.

19.1 Maintenance Points

The primary CFM protocol shims are called Maintenance Points (MPs). (See 19.6 for the other CFM protocol shim.) An MP can be either a MEP or an MHF. (See also 19.5 for a third type of CFM shim.) A MEP or an MHF each have two principle SAPs, an Active SAP, through which CFM Protocol Data Units (CFM PDUs) produced and consumed by the MP are exchanged, and a Passive SAP, through which data frames and unprocessed CFM PDUs pass. Figure 19-1 illustrates the principle SAPs, and the symbols used for MEPs, MIPs, and MHFs in other figures, e.g., Figure 22-1. Note that an MP can be a Down MP or an Up MP, depending on whether the Passive SAP is above or below the Active SAP, respectively, in a shim diagram.

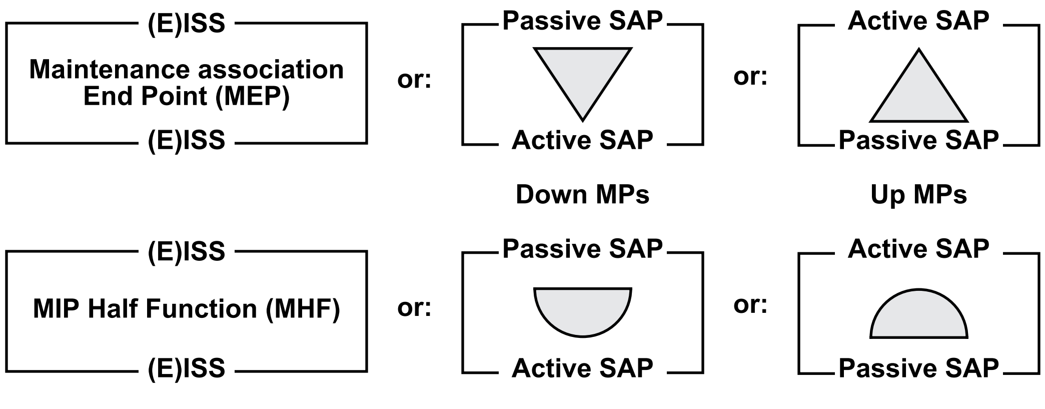

Figure 19-1—CFM Protocol shims

NOTE—The difference between Up MPs and Down MPs is explained more fully in 19.2.3 and 22.1.1.

19.2 Maintenance association End Point

A MEP can be created in either a Bridge Port (22.2) or an end station attachment to a LAN.

19.2.1 MEP identification

A MEP is associated with exactly one MA. It is identified for management purposes by two parameters, although other, additional parameters are used to manage the MEP:

a) The Maintenance Association Identifier (MAID), identifying the MA; and
b) A Maintenance association End Point Identifier (MEPID) specific to this MEP.

The MEP is identified for data forwarding purposes by:

c) A set of VIDs, including a Primary VID, inherited from the MA.

From the MA, the MEP inherits a Maintenance Domain, and from this it inherits:

d) A Maintenance Domain Level (MD Level), inherited from its Maintenance Domain; and
e) A Primary VID, inherited from its MA.

A MEP is associated with two Bridge Ports. For a Down MEP, both associations are with the same Bridge Port. For an Up MEP, the two associations can be with either the same, or with different, Bridge Ports (see 19.4, J.6). The two Bridge Ports are:

f) The Bridge Port on which the MEP is configured; and
g) The Bridge Port on which the MEP is implemented, and from which it inherits its Individual MAC address.

The set of MEPs configured with identical values for MAID define an MA. Thus, one can say that the MA, and not the MEPs, possesses this parameter. However, an MA is a diffuse entity that can be spread among a number of geographically separated Bridges. Each Bridge has its own Maintenance Association managed object for an MA, from which its MEP's MAID, MD Level, and Primary VID are derived. The selection of which of the MA's VIDs is the Primary VID can be overridden for a specific MEP. Although this information can be incorrectly configured (i.e., configured differently than some other Bridge), the Maintenance Association managed object ensures the uniform configuration of MEPs in a single Bridge.

Each individual MEP is configured with a MEPID that is unique within that MA. Again, the Maintenance Association managed object helps to maintain uniqueness, as it ties together the configuration parameters of the MEPs in a single Bridge and the same MA and ensures the uniqueness of the MEPIDs, at least in that Bridge.

The MAID can take a number of forms (see 21.6.5.1), including a Domain Name based string. To prevent incorrect operation in the face of unanticipated connections among systems, the MAID is unique over the domain for which CFM is to provide such protection. If the MAID is globally unique, then that domain is global. CFM can detect connectivity errors only over a set of MEPs in which MAIDs are unique.

The MEPID is an integer in the range 1–8191. It provides each MEP with the unique identity required for the MEPs to verify the correct connectivity of the MA.

NOTE—The range of the MEPID has been chosen to be large enough to accommodate most network needs, but small enough to be used as an index into a table of remote MEPs, and not require a table lookup, in a MEP Continuity Check Receiver (19.2.8) implemented in hardware.

19.2.2 MEP functions

The MEP:

a) Periodically transmits Continuity Check Messages (CCMs) according to the managed objects defined in item e) in 12.14.6.1.3;
b) Validates received CFM PDUs as specified in 20.1, 20.2, 20.3, and 20.46;
c) Discards those CFM PDUs at a lower MD Level to that configured in the MEP (20.7.1);
d) Transmits Loopback Messages (LBMs) and receives Loopback Replies (LBRs) as defined in 20.2, according to the managed objects defined in 12.14.7.3;
e) Transmits Linktrace Messages (LTMs), and transmits and receives Linktrace Replies (LTRs) as defined in 20.3, according to the managed objects defined in 12.14.7.4;
f) Responds to LBMs with LBRs as defined in 20.2.2;
g) Responds to LTMs with LTRs as defined in 20.3.2;
h) Maintains the MEP CCM Database as defined in 20.1.3;
i) May maintain the MIP CCM Database as defined in 20.1.3; and
j) Maintains the variables in 20.9.

Although defined as having either ISS or EISS interfaces, a MEP is highly constrained in its use of the EISS. Specifically:

k) Every frame passing through the MEP, in either direction, is emitted with its vlan_identifier, canonical_format_indicator, and drop_eligible parameters unchanged from the value received.
l) All PDUs generated by the MEP, namely CCMs, LBMs, LBRs, LTMs, and LTRs, use the MEP's configured Primary VID as the vlan_identifier parameter.
m) The canonical_format_indicator emitted on any frame is the appropriate value corresponding to the MAC attached to the Bridge Port.
n) The rif_information parameter is not present on any emitted frame.

19.2.3 MEP architecture

Figure 19-2 illustrates the component entities of a MEP. Entities that simply pass, redirect, or filter frames without altering them, and which have no internal state, are shown with dashed outlines; those that generate or transform frames, or that have an internal state, are shown with continuous outlines. State machines describing the latter entities are specified in Clause 20. Figure 19-2 shows that the two enclosing SAPs are not equivalent. The upper SAP in the figure is the Passive SAP. It lies closer to the DoSAP of the monitored service instance. The lower SAP in the figure is the Active SAP, the one that faces the monitored service instance. Thus, Figure 19-2 illustrates one of the Down MEPs shown in Figure 22-1 on page 208. To illustrate one of the Up MEPs of Figure 22-1, Figure 19-2 would have to be inverted. Note also in the diagram that there are two paths, one of which is present only in the Up MEP, and one of which is present only in the Down MEP.

19.2.4 MP Type Demultiplexer

There are two MP Type Demultiplexers in a MEP, the Active and the Passive Type Demultiplexers, and one in an MHF. All have the same purpose: to separate the data frames from those carrying CFM PDUs using the CFM encapsulation (21.2), output non-CFM frames to one output, and output the CFM frames to a second output. Frames containing CFM PDUs are sent to the Level Demultiplexer, and non-CFM frames are passed through the MEP intact.

NOTE—The model used for functional entities in this standard assumes that the operations of the components shown in Figure 19-2 and Figure 19-3 are instantaneous, so that the demultiplexing and multiplexing of frames, e.g., of CFM and non-CFM frames, cannot change the order of delivery of these frames. Although such reordering has no adverse consequences on this standard, and although it would not normally violate the frame ordering requirements of 6.3.3 (because CFM and non-CFM frames would seldom have the same source and destination MAC address pairs), this reordering could cause the Packet Loss Measurement operations of ITU-T Y.1731 (2006) to fail to operate correctly. IEEE Std 802.3, Clause 43, also can disrupt the operation of ITU-T Y.1731 Packet Loss Measurement operations.

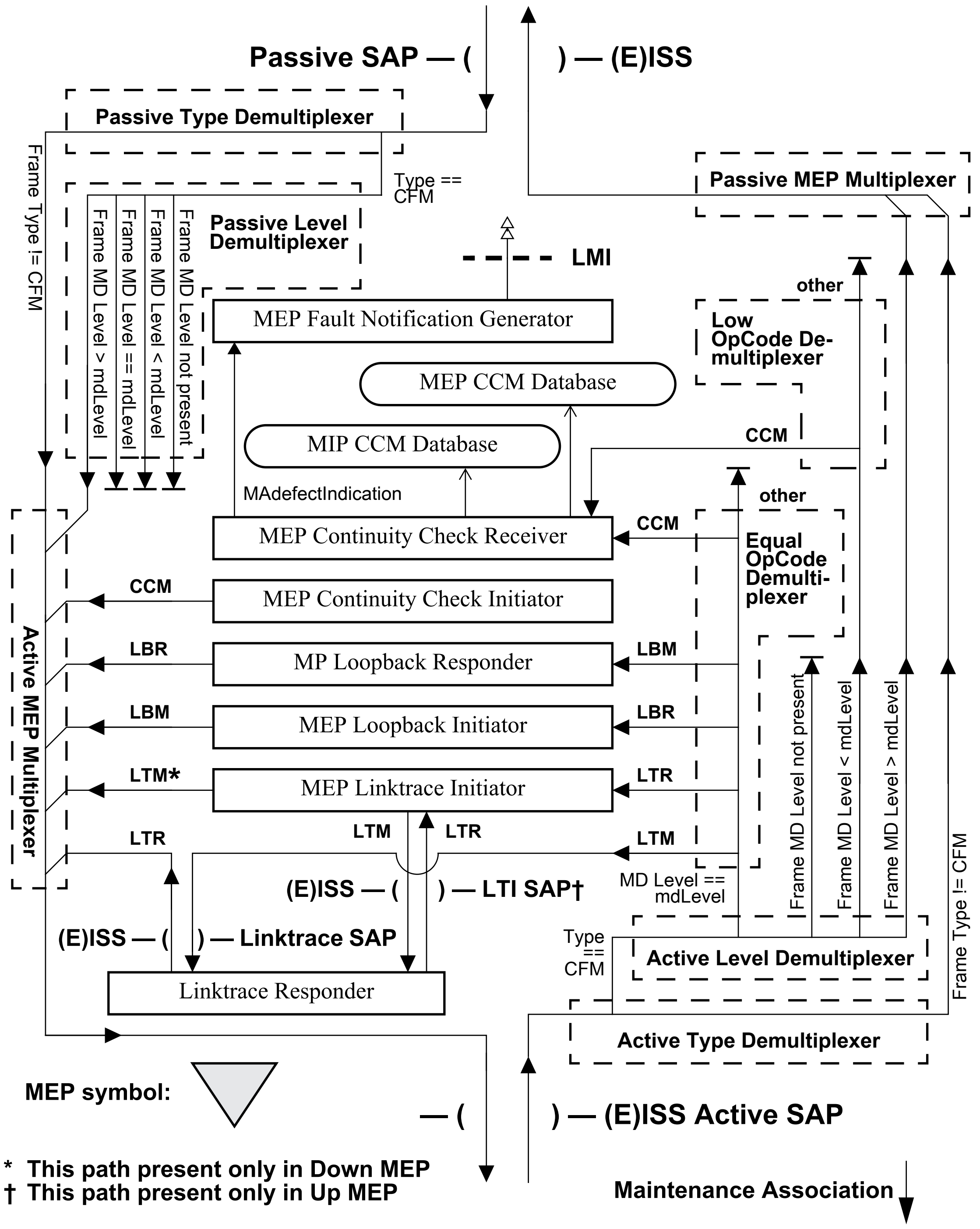

Figure 19-2—Maintenance association End Point (MEP)

19.2.5 MP Multiplexer

There are two MP Multiplexers in a MEP or in an MHF, the Active and the Passive Multiplexers. An MP Multiplexer merges frames from multiple inputs and outputs them to a single output. No frame queuing is specified or implied by this standard.

19.2.6 MP Level Demultiplexer

There are two MP Level Demultiplexers in a MEP, the Active and Passive Level Demultiplexers. Both separate CFM PDUs according to the MD Level contained within the PDU, and the MD Level configured in the MEP, and output them to different outputs. Specifically:

a) Any CFM PDU received that is too short to contain an MD Level header field shall be discarded.
b) Otherwise, the MD Level header field of the CFM PDU is compared to the MD Level configured for the MP [item b) in 12.14.5.1.3, item b) in 12.14.3.2.2, 20.7.1], and the CFM PDU is directed to one of the three outputs of the MP Level Demultiplexer, according to whether the CFM PDU's MD Level is less than, equal to, or greater than the MP's MD Level.

19.2.7 MP OpCode Demultiplexer

There are two MP OpCode Demultiplexers in a MEP, the Equal and the Low OpCode Demultiplexers, and one in an MHF. Each MP OpCode Demultiplexer separates CFM PDUs that carry different values in their OpCode fields (see 21.4.3), and either discards them or passes them to the appropriate state machine. Those PDUs that do not match any of the OpCodes recognized by an MP OpCode Demultiplexer, or that are too short to contain an OpCode, are discarded by either of a MEP's MP OpCode Demultiplexers, and passed to the Passive MHF Multiplexer by an MHF's MP OpCode Demultiplexer. Those PDUs that do match a recognized OpCode are stored in a variable and cause another Boolean variable to be set, according to Table 19-1. These variables, in turn, trigger state machine actions. The three MP OpCode Demultiplexers separate different sets of OpCodes; see Figure 19-2 and Figure 19-3.

Table 19-1—Actions taken by MP OpCode Demultiplexers

OpCode	MEP Equal OpCode Demultiplexer	MEP Low OpCode Demultiplexer	MHF OpCode Demultiplexer
CCM	Sets CCMreceivedEqual and CCMequalPDU for 20.18	Sets CCMreceivedLow and CCMlowPDU for 20.18	Sets MHFrecvdCCM and MHFCCMPDU for 20.15 and transfers CCM to Passive MHF Multiplexer
LBM	Sets LBMreceived and LBMPDU for 20.27	Discards PDU	Sets LBMreceived and LBMPDU for 20.27 and transfers LBM to Passive MHF Multiplexer
LBR	Sets LBRreceived and LBRPDU for 20.32	Discards PDU	Transfers LBR to Passive MHF Multiplexer
LTM	Transfers LTM to MEP Linktrace SAP	Discards PDU	Transfers LTM to MHF Linktrace SAP
LTR	Sets LTRreceived and LTRPDU for 20.40	Discards PDU	Transfers LTR to Passive MHF Multiplexer
Other	Discards PDU	Discards PDU	Transfers PDU to Passive MHF Multiplexer

19.2.8 MEP Continuity Check Receiver

The MEP Continuity Check Receiver receives CCMs whose MD Levels are equal to or less than the MD Level of the MEP. The MEP Continuity Check Receiver processes those at or lower than its own MD Level, regardless of destination_address, to update the MEP CCM Database. The MEP Continuity Check Receiver maintains one instance of the Remote MEP state machine (20.20) for each other MEP configured for this MA. In addition, the MEP Continuity Check Receiver maintains a single instance each of the Remote MEP Error state machine (20.22) and the MEP Cross Connect state machine (20.24).

If a MEP's Continuity Check Receiver state machine's allRMEPsDead variable (20.9.4) is set, and if the MEP is not associated with any VID and is a Down MEP (i.e., in position 5 in Figure 22-9), then the MAC_Operational parameter presented to the Passive SAP by the MEP is false, whatever the state of the MAC_Operational parameter received from the Active SAP. Thus, a completely failed MA causes the attached Bridge Port to appear as a failed link.

Optionally, the MEP Continuity Check Receiver may maintain the MIP CCM Database as described in 19.3.10.

Various error conditions, including the receipt of CCMs at a lower MD Level than that of the MEP, cause the Remote MEP state machine to detect a defect that can in turn generate a Fault Alarm.

19.2.9 MEP Continuity Check Initiator

The Continuity Check Initiator transmits Continuity Check Messages (CCMs) at the MD Level of the MEP only. The state machine for the Continuity Check Initiator is described in 20.12.

19.2.10 MP Loopback Responder

The Loopback Responder receives LBMs at the MD Level of the MP only, validates and filters them according to destination_address (19.4), and transmits LBRs in response. The state machine for the MP Loopback Responder is described in 20.27.

19.2.11 MEP Loopback Initiator

The Loopback Initiator transmits LBMs at the MD Level of the MEP only, and validates, filters according to destination_address (19.4), and counts the LBRs returned by other MPs' Loopback Responders. The state machine for the Loopback Initiator is described in 20.30.

19.2.12 MEP Linktrace Initiator

The Linktrace Initiator transmits LTMs at the MD Level of the MEP only, and validates, filters according to destination_address (19.4), and catalogs the LTRs returned by other Bridges' Linktrace Responders. The state machine for the Linktrace Initiator is described in 20.40. In an Up MEP the Linktrace Initiator transmits its LTMs to the Bridge's Linktrace Responder. In a Down MEP, the Linktrace Initiator transmits its LTMs through the MEP's own Active Multiplexer towards the MEP's Active SAP.

19.2.13 MEP LTI SAP

An Up MEP, but not a Down MEP, has a MEP LTI SAP, an instance of the ISS or EISS. This SAP is used to:

a) Transmit LTMs originated by the MEP's Linktrace Initiator; and
b) Receive LTRs from the Linktrace Responder.

19.2.14 MEP Linktrace SAP

Every MEP has a MEP Linktrace SAP, an instance of the ISS or EISS. This SAP is used to relay received LTMs to the Linktrace Responder.

NOTE—Both the MEP LTI SAP and MEP Linktrace SAP are required so that the MEP knows, when an LTR is returned to it by the Linktrace Responder, whether the LTR is a response to an LTM generated by the MEP and thus to direct it to the MEP Linktrace Initiator, or whether the LTR is a response to an LTM previously received by the MEP and thus to forward it out the Active SAP.

19.2.15 MEP CCM Database

The MEP CCM Database is maintained by the Remote MEP state machines (20.20), one entry for each of the other MEPs configured in the MA. In addition to the Remote MEP variables (20.19) used by the Remote MEP state machine, each entry holds information obtained from the remote MEP’s CCMs as listed in 12.14.7.6.3. See also 20.1.3.

19.2.16 MEP Fault Notification Generator

The Fault Notification Generator uses the MAdefectIndication variable (see 20.9.3) in order to generate a Fault Alarm to the administrator of the Maintenance Domain to which the MA is attached. This can be accomplished by sending an SNMP Notification. The state machine for the Fault Notification Generator is described in 20.35.

19.3 MIP Half Function

A MIP consists of two MIP Half Functions (MHFs) on a single Bridge Port, an Up MHF and a Down MHF. (See J.6 for a more detailed explanation.) An MHF may maintain a MIP CCM Database, separate from the MEP CCM Databases.

19.3.1 MHF identification

Every MHF is characterized by one configuration parameter for management purpose, and all of the MHFs of a MIP have identical values for that configuration parameter:

a) The Maintenance Domain to which the MHF is assigned.

For data flow purposes, the MHF is characterized by:

b) A Maintenance Domain Level (MD Level), inherited from its Maintenance Domain or from the Default MD Level managed object; and
c) A set of Primary VIDs, one from each MA associated with the MHF’s Maintenance Domain.

NOTE—When multiple VIDs are used for a single service instance, an MHF is aware of each MA’s VID set and Primary VID. Unlike the MEP, however, the MHF has no occasion to use a MAID, so is not identified by a MAID.

An MHF is associated with two Bridge Ports. For a Down MHF, both associations are with the same Bridge Port; for an Up MHF, the two associations can be with either the same, or with different, Bridge Ports (see 19.4, J.6). The two Bridge Ports are:

d) The Bridge Port on which the MHF is configured; and
e) The Bridge Port on which the MHF is implemented and from which it inherits its Individual MAC address.

19.3.2 MHF functions

Each MHF:

a) Validates received LTMs as specified in 20.1, 20.2, 20.3, and 20.46;
b) Optionally, validates received CCMs and builds the MIP CCM Database;
c) Responds to LBMs with LBRs as defined in 20.2.2; and
d) Responds to LTMs by forwarding them and responding with LTRs as defined in 20.3.2.

Although defined as having either ISS or EISS interfaces, an MHF is highly constrained in its use of the EISS. Specifically:

e) Every frame passing through the MHF, in either direction, is emitted with its vlan_identifier, canonical_format_indicator, and drop_eligible parameters unchanged from the value received.
f) All LBMs forwarded by the MHF carry the same vlan_identifier, canonical_format_indicator, and drop_eligible parameters as the received LBM.
g) All PDUs generated by the MHF, namely LBRs, and LTRs, are transmitted with a vlan_identifier equal to the Primary VID of the MA to which the vlan_identifier of the received LBM or LTM belongs.
h) The canonical_format_indicator emitted on any frame is the appropriate value corresponding to the MAC attached to the Bridge Port.
i) The rif_information parameter is not present on any emitted frame.

19.3.3 MHF architecture

Figure 19-3 illustrates a single MHF. Entities that simply pass, redirect, or filter frames without altering them, and which have no internal state, are shown with dashed outlines; those that generate or transform frames, or that have an internal state, are shown with continuous outlines. State machines describing the latter entities are specified in Clause 20. Figure 19-3 shows that the two enclosing SAPs are not equivalent. The upper SAP in the figure is the Passive SAP. This lies nearer an ISAP for the monitored service instance. The lower SAP in the figure is the Active SAP, the one through which CFM PDUs are actively transmitted and received. Thus, Figure 19-3 illustrates one of the Down MHFs shown in Figure 22-1. To illustrate one of the Up MHFs of Figure 22-1, Figure 19-3 would have to be inverted.

19.3.4 MHF Level Demultiplexer

The MHF Level Demultiplexer is identical to the MP Level Demultiplexer described in 19.2.6.

19.3.5 MHF Type Demultiplexer

The MHF Type Demultiplexer is identical to the MP Type Demultiplexer described in 19.2.4.

19.3.6 MHF OpCode Demultiplexer

An example of the MP OpCode Demultiplexer described in 19.2.7. The MHF OpCode Demultiplexer is fully described in that subclause.

NOTE—As indicated by the split arrows labeled “CCM” and “LBM” in Figure 19-3, some of the frames output from the MHF OpCode Demultiplexer are supplied to two other entities, and not just one.

19.3.7 MHF Multiplexer

The MHF Multiplexer is identical to the MP Multiplexer described in 19.2.5. There are two MHF Multiplexers in an MHF, the Active MHF Multiplexer and the Passive MHF Multiplexer.

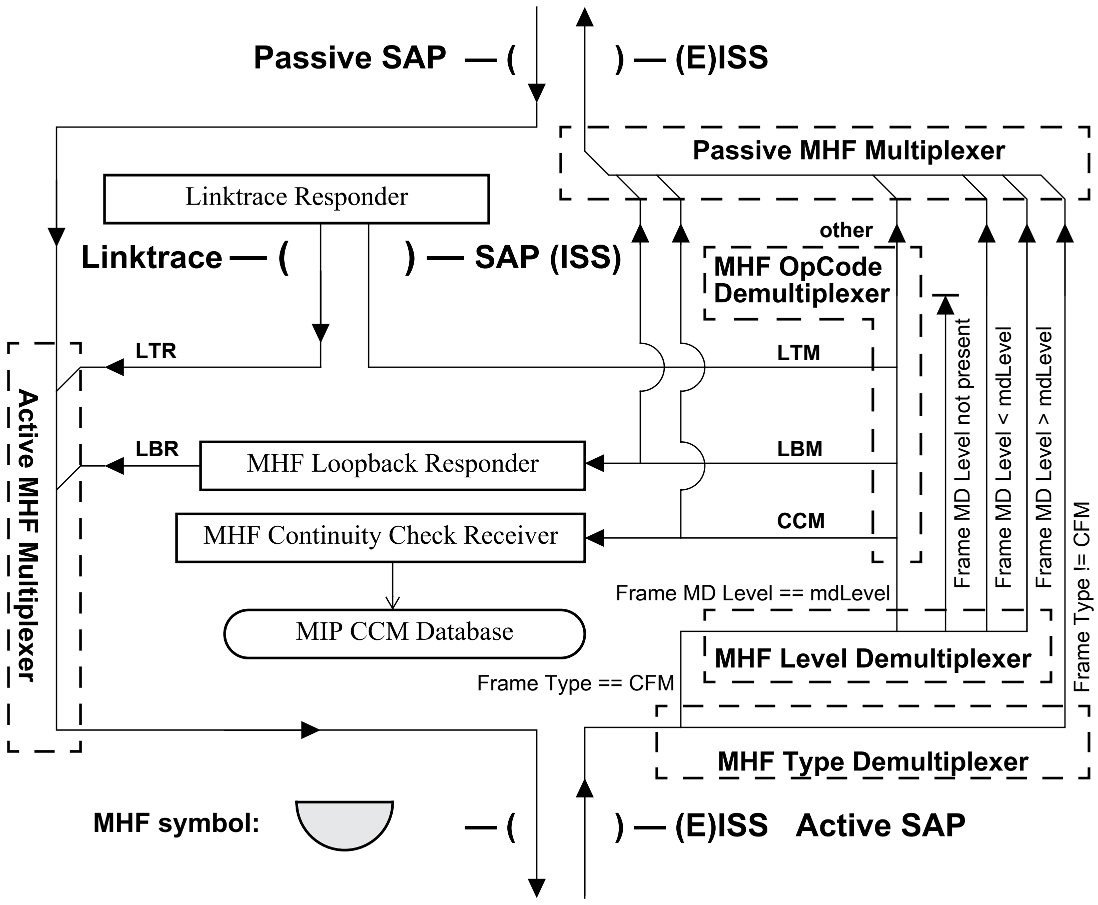

Figure 19-3—MIP Half Function

19.3.8 MHF Loopback Responder

The MHF Loopback Responder is identical to the MP Loopback Responder described in 19.2.10.

19.3.9 MHF Continuity Check Receiver

The optional MHF Continuity Check Receiver receives CCMs, validates them, filters them by destination_address parameter (19.4), and contributes to the MIP CCM Database. From Figure 19-3, we can see that all CCMs at the same MD Level as the MHF are presented to the MHF Continuity Check Receiver. No other information in the received CCMs is examined. In particular, the MAID in the CCM is not compared to the MHF's MAID.

The MHF Continuity Check Receiver, and the MIP CCM Database it maintains, are not required for conformance to this standard.

19.3.10 MIP CCM Database

The MIP CCM Database is optionally maintained by an MHF or MEP. It is a list of every distinct {FID, source_address, Port number} triple from all of the CCMs presented to all of the Bridge's MPs' Continuity Check Receivers since the Bridge was last reset. Entries in the MIP CCM Database are timed out very slowly, on the order of one day, so that their information is available for fault isolation long after the information is no longer used for frame forwarding.

19.3.11 MHF Linktrace SAP

Every MHF has an MHF Linktrace SAP, an instance of the ISS or EISS. This SAP is used to relay received LTMs to the Linktrace Responder, and to pass LTRs from Linktrace Responder to the MHF's Active SAP.

19.4 Maintenance Point addressing

The CFM entities within an MP recognize the Group addresses for CCM and LTM PDUs listed in Table 8-9 and Table 8-10. In addition, they recognize the Individual MAC address of the Bridge Port on which the MP is operating. This is the Bridge Port on which the MP is configured, if the Bridge created the MP using the Individual MP address model (J.6.1), and is a CFM Port, if the Bridge created the MP using the Shared MP address model (J.6.2).

A Down MP shall not be assigned an Individual MAC address [item i) in 12.14.7.1.3] that is the same as the Individual MAC address configured for an MP on any other Bridge Port, because if two Bridge Ports are connected to the same LAN, but have the same MAC address, then both would respond to an LBM sent to that MAC address. The CFM protocol is specifically designed to allow an Up MP to be assigned an Individual MAC address [item i) in 12.14.7.1.3] that is the same as the Individual MAC address configured for an Up MP on some other Bridge Port, because it is impossible to tell, from outside the Bridge, from which Up MP a given CFM PDU was actually transmitted, or to which physical entity a given CFM PDU was delivered. See J.6 for a discussion of the assignment of Individual MAC addresses to MPs.

All of the CFM entities within one MP use the same CCM and LTM addresses, and Individual MAC address. The various multiplexing and demultiplexing entities pay no attention to the destination_address parameter of a received CFM PDU. Only the entities responsible for processing received CFM PDUs check a received frame's destination_address parameter. Each of these entities recognizes frames whose destination_address parameters match one or more of the MAC addresses as shown in Table 19-2. Each of these entities filters and does not process frames that match none of the specified addresses.

Table 19-2—Received destination_address filtering by CFM entities

Entity	PDU	Received frame destination_address comparison		
		Individual address	CCM Group address	LTM Group address
MEP Continuity Check Receiver (19.2.8)	CCM	No address matching performed.		
MP Loopback Responder (19.2.10)	LBM	Match	Equal MD Level	Equal MD Level
MEP Loopback Initiator (19.2.11)	LBR	Match	Not used	Not used
MEP Linktrace Initiator (19.2.12)	LTR	Match	Not used	Not used
MHF Loopback Responder (19.3.8)	LBM	Match	Not used	Equal MD Level
MHF Continuity Check Receiver (19.3.9)	CCM	Match	Equal MD Level	Not used
Linktrace Responder (19.6)	LTM	Match	Not used	Equal MD Level

NOTE—Table 19-2 and the definitions of the various CFM receiving entities are more liberal in recognizing certain Individual or Group MAC addresses than the corresponding definitions of the CFM transmitting entities. For example, the MEP Continuity Check Receiver recognizes unicast CCMs, even though the MEP Continuity Check Initiator can only transmit multicast CCMs. This allows for future enhancement of CFM and for improved compatibility with ITU-T Y.1731 (2006).

19.5 Linktrace Output Multiplexer

The third type of CFM protocol shim is the Linktrace Output Multiplexer (LOM). An LOM has two principle SAPs, an Active SAP, through which CFM PDUs are transmitted, and a Passive SAP, through which pass data frames and unprocessed CFM PDUs. Figure 19-4 illustrates these SAPs and the symbol used for the LOM in other figures, e.g., Figure 22-1. An LOM can be only a Down LOM; its Passive SAP is always above the Active SAP in a shim diagram.

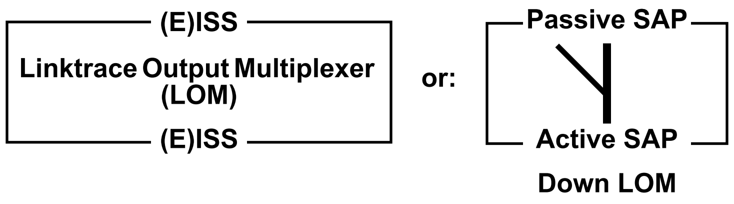

Figure 19-4—Linktrace Output Multiplexer shim

As illustrated in Figure 19-5, the LOM has only a single active entity, the LOM Multiplexer. It injects LTMs from the Linktrace Responder among the frames passing from the LOM's Passive SAP to its Active SAP. Frames received from the LOM's Linktrace SAP are transmitted through only the Active SAP, and not through the Passive SAP. All frames received on either the Active or Passive SAP are passed through the LOM unchanged, and only to the Passive or Active SAP, respectively.

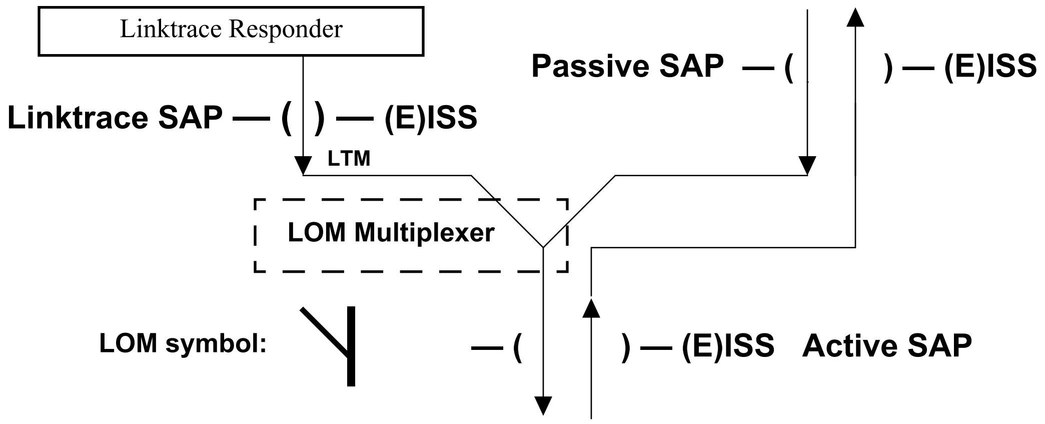
Linktrace Responder
Passive SAP — ( ) — (E)ISS
Linktrace SAP — ( ) — (E)ISS
LTM
LOM Multiplexer
LOM symbol:
— ( ) — (E)ISS Active SAP

Figure 19-5—Linktrace Output Multiplexer architecture

NOTE 1—An LOM is not a Maintenance Point, but an entity that assists the Linktrace Responder (19.6), also not a Maintenance Point.

NOTE 2—The LOM exists on every Bridge Port in order to allow the Linktrace Responder to transmit an LTM on a particular Bridge Port without the possibility of that LTM being reflected back towards the Bridge's Forwarding process. However, the LOM has no constituent entity that differentiates CFM frames from any other frames.

19.6 Linktrace Responder

A single Linktrace Responder serves all of the MHFs and MEPs in a Bridge. It is connected to every MP and LOM in the Bridge, via the Linktrace SAPs and LTI SAPs in those MPs and LOMs.

NOTE—A Linktrace Responder is not a Maintenance Point, but an entity that assists the Maintenance Points (MEP and MHF, 19.2 and 19.3) in performing their functions.

In order to minimize the possibility of generating an unbounded number of responses to an LTM, an LTM is always transmitted from a Bridge on a single Bridge Port (20.3). If the LLC entity of a Bridge Port (see Figure 22-1) were used to transmit an LTM on a particular Bridge Port, the Bridge Port connectivity entity (8.5.1) would direct the LTM to the LAN, but it would also pass that frame up towards the Frame filtering entity (8.6.3), as it does for BPDUs. BPDUs are never transmitted to other Bridge Ports, however, because the Frame filtering entity filters them, based on their destination MAC addresses. The placement of MHFs at different MD Levels on different Bridge Ports, all for the same VLAN, precludes filtering LTMs in this manner. Therefore, the Linktrace Responder filters incoming LTMs by the destination_address parameter (19.4), and a path is required that will direct an LTM only down the Bridge Port, towards the LAN, and not towards the Frame filtering entity.

This path is provided by the Linktrace SAPs in the MEP (Figure 19-2), the MHF (Figure 19-3), and the LOM (Figure 19-5). The Linktrace SAPs are also used to direct a received LTM to the Linktrace Responder when the LTM first encounters an MP at the appropriate MD Level.

Referring to Figure 22-1, we can see that an LTM originated by a Down MEP Linktrace Initiator is directed out to the LAN and is thus output on a single Bridge Port, as desired. The situation is different for an LTM originated by an Up MEP. The Bridge in which an Up MEP resides is, in effect, the first hop of the Linktrace protocol. The MEP LTI SAP is required in order to provide a path through which the MEP can direct its originated LTM to the Linktrace Responder. This path can also be used by the Linktrace Responder to return the resultant LTR to the MEP Linktrace Initiator. See Figure 20-13 for a diagrammatic explanation of these paths.

Table 19-3 summarizes the SAP connections between the MPs and LOM on the one hand, and the Linktrace Responder, on the other.

Table 19-3—SAP use for Linktrace Messages and Linktrace Replies

Entity taking action		Sends originated LTM to	Relays received LTM to	Sends to Active SAP an LTR received on	Sends to Active SAP an LTM received on
Up	MEP	MEP LTI SAP	MEP Linktrace SAP	MEP Linktrace SAP	Not applicable [a]
	MHF	Not applicable [b]	MHF Linktrace SAP	MHF Linktrace SAP	Not applicable [a]
Down	MEP	Not applicable [c]	MEP Linktrace SAP	MEP Linktrace SAP	Not applicable [a]
	MHF	Not applicable [b]	MHF Linktrace SAP	MHF Linktrace SAP	Not applicable
	LOM	Not applicable [b]	Not applicable [d]	Not applicable[a]	LOM Linktrace SAP

[a] The Linktrace Responder only forwards LTMs through an LOM.
[b] The MHF and LOM have no MEP Linktrace Initiator, so cannot originate LTMs.
[c] The Down MEP transmits the LTMs it originates through its Active SAP.
[d] The LOM is not an MP; it cannot detect received LTMs.

A single instance of the LTR Transmitter state machine is instantiated by the Linktrace Responder. It is described in 20.45.

Insert a new Clause 20 as follows:

20. Connectivity Fault Management protocols

Maintenance association End Points (MEPs) and Maintenance association Intermediate Point (MIPs) can participate in the following CFM Protocols:

a) Continuity Check protocol (20.1);
b) Loopback Protocol (20.2); and
c) Linktrace protocol (20.3).

The detailed operation of these protocols by the MEP (19.2) and MIP Half Function (MHF, 19.3) components introduced in Clause 19 is specified in terms of a number of state machines, and the variables and procedures associated with each machine. The state machines defined adhere to the conventions in IEEE Std 802.1Q-2005, 13.19, and Figure 13-10.

This clause specifies the following:

d) Relationships among the various CFM state machines and their variables (20.4);
e) Requirements for decrementing the timer counters used by a number of CFM state machines (20.5);
f) Maintenance Domain variables (20.7) that control all Maintenance Associations belonging to the Maintenance Domain;
g) Maintenance Association variables (20.8) that control all MEPs and MHFs belonging to the Maintenance Association;
h) MEP variables (20.9) that control the MEP's overall operation, and either apply to the majority of state machines or facilitate communication of information between the individual MEP state machines;
i) MEP Continuity Check Initiator (19.2.9) variables (20.10), procedures (20.11), and state machine (20.12);
j) (optional) MHF Continuity Check Receiver (19.3.9) variables (20.13), procedures (20.14), and state machine (20.15);
k) MEP Continuity Check Receiver (19.2.8) variables (20.16), procedures (20.17), and state machine (20.18);
l) Remote MEP variables (20.19) and state machines (20.20) used by the MEP Continuity Check Receiver to track each Remote MEP;
m) Remote MEP Error variables (20.21) and the Remote MEP Error state machine (20.22) used by MEP Continuity Check Receiver to track received invalid CCMs;
n) MEP Cross Connect variables (20.23) and state machine (20.24) used by the MEP Continuity Check Receiver to track received CCMs that could indicate a cross connect defect;
o) MP Loopback Responder (19.2.10) variables, (20.25), procedures (20.26), and state machine (20.27);
p) MEP Loopback Initiator (19.2.11) variables (20.28), transmit procedures (20.29), and state machine (20.30), and receive procedures (20.31), and state machine (20.32);
q) MEP Fault Notification Generator (19.2.16) variables (20.33), procedures (20.34) and state machine (20.35);
r) MEP Linktrace Initiator (19.2.12) variables (20.36) and transmit procedures (20.37), and receive variables (20.38), procedures (20.39), and state machine (20.40);
s) Linktrace Responder variables (20.41), procedures (20.42), and state machine (20.43), used by the Linktrace Responder (19.6); and
t) LTR Transmitter state machine (20.45) used only by the Linktrace Responder.

To facilitate extension of this standard while maximizing interoperability, this clause further specifies:

u) Rules for CFM PDU validation and versioning (20.46) that support specific goals for extensibility and interoperability (20.46.1);
v) Rules for the identification of incoming CFM PDUs (20.47); and
w) Rules for the use of transaction identifier and sequence number fields (20.48).

The models of operation in this clause provide a basis for specifying the externally observable behavior of MHFs and MIPs, and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified. Conformance of equipment to this standard is purely in respect of observable protocol.

NOTE—Clause 18 introduces the principles of CFM operation and the network architectural concepts that support it. Clause 19 breaks down the CFM protocol entities into their components. Clause 21 specifies the PDU formats used by the CFM protocols. The use of CFM within systems and networks is further described in Clause 22.

20.1 Continuity Check protocol

The Continuity Check protocol is performed by the MEP Continuity Check Initiator (19.2.9), the MEP Continuity Check Receiver (19.2.8) of the MEP, and optionally, in the MHF Continuity Check Receiver (19.3.9) of the MHF.

For the purposes of Fault detection and the Continuity Check protocol, we can define an MA as “a set of MEPs, all of which are configured with the same MAID and MD Level, each of which is configured with a MEPID unique within that MAID and MD Level, and all of which are configured with the complete list of MEPIDs.” The Continuity Check Message (CCM, 21.6) provides a means to detect connectivity failures in an MA. Each MEP can be configured to periodically transmit a CCM. A connectivity failure is then defined as either:

a) Inability of a MEP to receive three consecutive CCMs from any one of the other MEPs in its MA, indicating either a MEP failure or a network failure;
b) Reception by a MEP of a CCM with an incorrect transmission interval, indicating a configuration error;
c) Reception by a MEP of a CCM with an incorrect MEPID or MAID, indicating a configuration error or a cross connect error;
d) Reception by a MEP of a CCM with an MD Level lower than that of the MEP, indicating a configuration error or a cross connect error; or
e) Reception by a MEP of a CCM containing a Port Status TLV or Interface Status TLV indicating a failed Bridge Port or aggregated port.

In order to take the fullest advantage of CFM, the operator defines “correct connectivity” as being the configuration of the MEPs. If an MA is defined for every service instance, then the Continuity Check protocol can detect 100% of all connectivity faults, cross connection errors, or misconfigurations that occur within the boundaries of the configured MAs.

The transmission rate for CCMs is configurable, to accommodate different needs for error reporting, and different capabilities of network devices. The loss of three CCMs can be detected in as little as 10.8 ms (see Table 21-16). The reception of a cross connected CCM is an instantaneous occurrence.

A MEP uses its own configured timer to detect the loss of CCMs, rather than using the transmission rate of the sender (signaled in the CCM), in order to minimize the complexity of a MEP tracking hundreds of other MEPs. The timer is reset when a valid CCM is received, and a defect declared if the timer expires. The timer value is chosen such that the loss of three consecutive CCMs from a given remote MEP triggers a defect for that remote MEP, and the timer accuracy chosen to ensure against a premature defect indication. In addition, state machines in the MEP track the receipt of CCMs from remote MEPs with unexpected information, CCMs from remote MEPs not configured in the receiving MEP, CFM PDUs from a lower MD Level, or CCMs from MEPs in another MA.

A CCM is carried in a multicast frame, with a destination_address parameter chosen according to the transmitting MEP's MD Level, according to Table 8-9. The CCM does not generate a response. Therefore, in a network with *n* MEPs, *n* CCMs need to be transmitted periodically, one from each MEP. Compared to a full mesh of point-to-point messages in terms of frames carried on wires, multicast CCMs require less bandwidth than the point-to-point messages, except of course for MAs with only two MEPs. And for each MEP, only one transmission and $n – 1$ receptions are required, instead of $n – 1$ of each. In addition, even in MAs with only two MEPs, using multicast instead of unicasts for CCMs obviates the need for configuration or discovery of those MEPs' MAC addresses.

Of more importance than frame count is the fact that making the CCM a multicast frame enables the detection of a cross connect error when MEPs from two different service instances are accidentally connected. This type of error can result from a wiring error or a misconfiguration of VID mapping parameters at the boundary between two Maintenance Domains. Were CCMs unicast to specific MAC addresses, then a merging of two service instances, e.g., the accidental concatenation of two point-to-point services into a 4-SAP LAN service, would go undetected; unicast CCMs would still be delivered properly. This scenario also points out why globally unique MAIDs are important.

Cataloging the receipt of CCMs ensures that a MEP, say, MEP 1, knows that it is receiving them from exactly the correct set of remote MEPs; it does not ensure that the CCMs transmitted by that same MEP are being received by the remote MEPs. For example, suppose that in a 10-MEP MA, all of MEP 1's CCMs are being lost. All of the other MEPs in the MA would know about MEP 1's problem, because they would not be receiving 1's CCMs. But, MEP 1 itself would be unaware of the problem. In order for the operator to decide whether an MA is working properly, every MEP in the MA would have to be inspected.

For this reason, the Remote Defect Indication (RDI), a single bit, is carried by the CCM. The absence of RDI in a CCM indicates that the transmitting MEP is receiving CCMs from all configured MEPs. In the case of MEP 1's unidirectional connectivity, MEP 1 would receive the RDI in the CCM from each remote MEP that is not receiving MEP 1's (or any other MEP's) CCMs. Thus, every MEP, including the receive-only MEP 1, is aware that a problem exists. The presence of RDI enables a system administrator to inspect any single MEP belonging to an MA, and discover whether there is any MEP in the MA that is detecting a defect, as well as which particular MEPs have defects.

A sequence number should be transmitted in every CCM. The receiver may use this sequence number to detect and count CCM losses. Detection of occasional CCM losses can enable a network administrator to notice suboptimal network conditions such as overloaded links, and correct them before connectivity defects and their consequent Fault Alarms occur.

20.1.1 MAC status reporting in the CCM

The CCM can carry information about the status of the Bridge Port and/or aggregated port on which the transmitting MEP is configured in the Port Status TLV (21.5.4) and Interface Status TLV (21.5.5). This information is not strictly within the scope of the Maintenance Association bounded by the MEPs transmitting and receiving the CCM. But, by providing Bridge Port and aggregated port status information in an Up MEP's CCMs, this TLV allows the detection of an error immediately outboard of the Maintenance Association. This in turn supports fault isolation where a problem has arisen at the point of connection to a service instance, rather than within the equipment supporting the service instance. The Port Status TLV and Interface Status TLV are particularly useful where the boundary between the equipment used by two operators to support a service instance is a link or connection that could fail. Use of the Port Status TLV and Interface Status TLV allows monitoring to be uninterrupted from end to end, without requiring management access by both operators to a single item of interface equipment, or relying on monitoring by the user of the concatenated service. The Port Status TLV and Interface Status TLV allow errors that are immediately outboard of the Maintenance Association to be distinguished easily from those properly attributed to the Maintenance Association itself.

Figure 18-7 provides an example. Consider a failure of the access link that connects the customer equipment (1) directly to Port b of Provider Bridge 2. Without the Interface Status TLV, none of the MEPs at the operator or service provider MD Level would be notified of the error. The Interface Status TLV allows the failure to be signaled in CCMs transmitted by the Port b service provider and operator MD Level MEPs.

20.1.2 Defects and Fault Alarms

Defects are separated from Fault Alarms (19.2.16), as is standard practice for service providers.[10] The loss of CCMs or the reception of a cross connected CCM are defects. A Fault Alarm is a Management operation (12.14.7.7), an unsolicited announcement by a Bridge. If the CFM MIB module is implemented (17.5), it is an SNMP Notification. It is issued when the MEP Fault Notification Generator state machine (20.35) detects that a configured time period (default, 2.5 s) has passed with one or more defects indicated, and Fault Alarms are enabled. The state machine can transmit no further Fault Alarms until it is reset by the passage of a configured time period (default, 10 s) during which no defect indication is present. The normal operator procedure upon receiving a Fault Alarm is to inspect the reporting MEP’s managed objects, diagnose the fault, correct the fault, examine the MEP’s managed objects to see whether the MEP Fault Notification Generator state machine has reset, and repeat those steps until the Fault Alarm has cleared.

A number of separate defects are maintained by a MEP, as shown in Table 20-1. The defects are ranked by priority. If a higher priority defect occurs after a lower priority defect has triggered a Fault Alarm, but before the Fault Alarm has reset, then the MEP will immediately issue another Fault Alarm. This enables the operator to reliably prioritize Fault Alarms. For example, cross connect errors are typically of greater concern in a service provider environment than loss of connectivity errors.

Table 20-1—Fault Alarm defects and priorities

Defect		**Priority**	
Variable	**highestDefect (20.33.9)**	**highestDefectPri (20.33.8)**	**Importance**
xconCCMdefect (20.23.3)	DefXconCCM	5	Most
errorCCMdefect (20.21.3)	DefErrorCCM	4	
someRMEPCCMdefect (20.33.5)	DefRemoteCCM	3	
someMACstatusDefect (20.33.6)	DefMACstatus	2	
someRDIdefect (20.33.7)	DefRDICCM	1	Least

Only the highest priority defect is reported in the Fault Alarm. Table 20-1 shows the relationships between the variables that indicate defects, the priorities of these defects, and the enumerated value reported in the Fault Alarm for each defect.

[10]See, for example, ITU-T G.806 (2006).

20.1.3 CCM reception

Every active MEP receives and catalogs CCMs in its MEP CCM Database. Every CCM shall be examined to be sure that its MAID matches that configured in the receiving MP. A receiving MEP checks to ensure that its own MEPID does *not* match that in the received CCM. (This could indicate either a duplicate MEPID or a network forwarding loop.) The information in the CCM is catalogued in the MEP CCM Database, indexed by the received MEPID. The information kept in the MEP CCM Database is listed in 12.14.7.6.3.

The MIP CCM Database is optionally maintained by a MEP Continuity Check Receiver or an MHF Continuity Check Receiver. It is a list of every distinct {FID, source_address, Port number} triple from all of the CCMs presented to all of the Bridge's Down MHFs' MHF Continuity Check Receivers since the Bridge was last reset. (See 8.8.7 for a discussion of VIDs and FIDs.) An entry in the CCM Database is retained for at least 24 h after the last CCM with its triple is received and is removed from the MIP CCM Database after, at most, 48 h. If the resources allocated for the MIP CCM Database are not sufficient to maintain all of the entries for the required time, then the least-recently updated entries should preferentially be removed to make room for new entries. See 20.3.2 for a description of the use of the MIP CCM Database.

20.2 Loopback protocol

A unicast Loopback Message (LBM) is used for Fault verification and isolation. To verify the connectivity between MEPs and MIPs, a MEP can be instructed by a system administrator to issue one or more LBMs. The LBM is initiated by a MEP with specified destination_address, priority, and drop_eligible parameters, the destination_address being the Individual MAC address of another MP within the same Maintenance Association as the transmitting MEP. The receiving MP responds to the LBM with a unicast Loopback Reply (LBR).

The Loopback protocol is performed by the MEP Loopback Initiator (19.2.11) and the MP Loopback Responder (19.2.10), the latter residing in either a MEP or an MHF.

20.2.1 Loopback Message transmission

LBMs are transmitted by operator command as specified in 12.14.7.3. Each LBM transmitted contains a Loopback Transaction Identifier field (21.7.3) that is incremented with each transmission. That enables the transmitting MEP to correlate the returned LBRs with the transmitted LBMs. The LBM can carry an arbitrary amount of data to help diagnose faults that are sensitive to the amount or pattern of data in a frame [item d) in 12.14.7.3.2]. It can be sent with any priority or drop_eligible parameters [item e) in 12.14.7.3.2].

No means for specifying the rate at which the LBMs are to be sent is provided. A Bridge shall not transmit LBMs at a rate that would cause the queues serving that Bridge Port to overflow and drop LBMs, were there no other traffic being inserted into those queues.

20.2.2 Loopback Message reception and Loopback Reply transmission

When an LBM is received by an MP Loopback Responder (19.2.10), it may be examined for validity and discarded if invalid. Whether or not the LBM is examined for validity, it shall be discarded if the source_address is a Group and not an Individual MAC address. Also, if the destination_address matches neither the MAC address of the receiving MP nor the Group MAC address from Table 8-9 appropriate to the MD Level of the receiving MP, the MP shall discard the LBM. If the receiving MP Loopback Responder resides in an MHF and the destination_address is a Group MAC address, the MHF shall discard the LBM. If the frame passes these tests, the receiving MP generates an LBR and transmits it to the originating MEP. Every M_UNITDATA.indication (or EM_UNITDATA.indication) parameter and octet in the mac_service_data_unit in the LBM is copied to the LBR's M_UNITDATA.request (or EM_UNITDATA.request) with the following exceptions:

a) The source_address parameter of the received LBM is used as the destination_address parameter for the transmitted LBR;
b) The source_address parameter for the LBR is the MAC address of the replying MP; and
c) The OpCode field is changed from LBM to LBR.

The receiver of an LBM shall not interpret any of the other fields or TLVs in the LBM. The contents of any TLVs that do not violate the validation criteria of 20.46, but are not specified in this standard, and any valid Organization-Specific TLV not known to the receiving MP, shall be ignored, and not interpreted, by the receiver, except that their contents shall be copied to the LBR.

NOTE—This standard provides no means for transmitting an LBM with a Group MAC address. The MP Loopback Responder responds to LBMs sent to the CCM Group address because ITU-T Y.1731 (2006) provides for transmitting such messages. See J.4.

20.2.3 Loopback Reply reception

When an LBR is received by an MHF (an anomalous occurrence), it is ignored, as the MIP has no receiving entity for an LBR. When an LBR is received by a MEP Loopback Initiator (19.2.11), it is checked to see whether the destination_address parameter matches the MAC address of the receiving MEP, and discarded if not. Next it is checked to see whether its Loopback Transaction Identifier field matches that of a recently transmitted LBM, and the appropriate counter incremented, either the in-sequence counter [item y) in 12.14.7.1.3] or the out-of-sequence counter [item z) in 12.14.7.1.3].

If the Shared MP address model for implementing Connectivity Fault Management is used by a particular Bridge, then as mentioned in 20.47, there is an ambiguity in the identification of the particular Up MP to which an LBR is destined. Therefore, a Bridge using this model uses a single variable for managing the Loopback Transaction Identifier fields for all MPs sharing the same MAC address, MAID, and MD Level.

The receiver of an LBR may compare its contents bit-for-bit with the remembered contents of the corresponding LBM, and increment a variable [item aa) in 12.14.7.1.3] if they do not match.

20.3 Linktrace protocol

The Linktrace protocol is performed by the MEP Linktrace Initiator (19.2.12), associated with a single MEP, and the Linktrace Responder (19.6) associated with a Bridge.

An LTM is transmitted by a MEP Linktrace Initiator in order to perform path discovery and fault isolation. The LTM carries a target MAC address as part of its payload. It is carried in a multicast frame, with a destination_address taken from Table 8-10 according to the MD Level of the transmitting MEP, and is relayed as such through the Bridged Network until it reaches an MP at the appropriate MD Level. That MP intercepts the LTM and deflects it to a Bridge's Linktrace Responder. The Linktrace Responder determines whether its Bridge's MAC Relay Entity would forward an ordinary data frame with the specified target MAC address to a single egress Bridge Port, or would filter or flood it. If the single egress port is found, or if the receiving MP is the terminating MP, the Linktrace Responder sends a unicast Linktrace Reply (LTR) to the originator of the LTM, whose MAC address was also carried as payload in the LTM. The LTR is sent after a random delay in the range $0 < \text{delay} \leq 1$ s, to mitigate the load on the originating MEP. In addition, if the MP through which the LTM was received was an MHF, the Linktrace Responder forwards an altered version of the LTM out of a single Bridge Port in the direction of the target MAC address. The MEP Linktrace Initiator that originated the initial LTM collects the LTRs. These provide sufficient information to construct the sequence of MPs that would be traversed by a data frame sent to the target MAC address.

Path discovery in a connectionless environment is more challenging than a connection-oriented environment. In Bridged LANs, it can be especially challenging, since the MAC address of the target of an LTM can be deleted from the Filtering Database by a TCN immediately after a network topology change, and ages out and is deleted a few minutes (Max Age) after last being learned when a fault results in isolating the target from the MEP originating the LTM.

CFM offers three ways to address the problem of ageing out MAC addresses:

a) Launching the Linktrace protocol automatically, immediately following the detection of a fault, such that it gets exercised within the Max Age window;
b) Maintaining information about the destination MP at the intermediate points along the path by using the optional MIP CCM Database, 19.3.9 and 20.1.3; and
c) Maintaining normal path information by issuing periodic LTMs during normal operations.

NOTE—Periodic LTMs allow a system administrator to determine the MIPs along the paths among MEPs during normal operations, thus establishing a baseline for subsequent fault isolation. However, triggering LTMs at rates approaching the CCM transmission interval could overwhelm the Bridges' ability to process CFM PDUs. See also 22.5.

20.3.1 Linktrace Message origination

LTMs are transmitted by a MEP Linktrace Initiator in response to an operator command (12.14.7.4). The nextLTMtransID [item b) in 12.14.7.4.3, 20.36.1] of each LTM transmitted is retained for at least 5 s after the LTM is transmitted. The LTM Transaction Identifier values transmitted by a given MEP are unique over this time period in order to match LTRs returned by slow MPs to the LTM that triggered the slow LTR. Initializing the nextLTMtransID to a random value[11] and incrementing it once per LTM transmitted satisfies this criterion. For each LTM transmitted, the MEP Linktrace Initiator creates an entry in a Linktrace database (20.36.2). This entry is indexed by nextLTMtransID for retrieval when a corresponding LTR is received.

The transmitting Bridge can maintain a separate sequence of LTM Transaction Identifiers per MEP if multiple MEPs in the same service instance and at the same MD Level have different MAC addresses, but uses a common sequence of LTM Transaction Identifiers for MEPs that share both of these characteristics.

The destination_address of an LTM is the Group MAC address reserved for LTMs and appropriate to the MD Level of the originating MEP according to Table 8-10.

The Target MAC Address field [item c) in 12.14.7.4.2, 21.8.6] in the LTM can be any Individual MAC address. However, since MEPs are required to periodically transmit CCMs, a MEP's MAC address is likely to be known to the MIPs along the path. Conversely, a MIP's MAC address, because the MIP does not initiate any CFM PDUs, can be unknown to intermediate Bridges unless it has recently replied to an LBM or LTM.

The MEP Linktrace Initiator in a Down MEP transmits the LTM through its Active SAP, towards the LAN attached to its Bridge Port. The MEP Linktrace Initiator in an Up MEP transmits the LTM through its MEP LTI SAP to its own Bridge's Linktrace Responder. That Linktrace Responder handles the LTM as described in 20.3.2. The Linktrace Responder can forward the LTM through the LOM Linktrace SAP and can respond through the MEP LTI SAP with an LTR. Thus, when an LTM is initiated in an Up MEP, the originating Bridge is the first hop.

20.3.2 Linktrace Message reception, forwarding, and replying

An LTM at a particular MD Level is forwarded as an ordinary multicast data frame until it encounters a MEP at an equal or higher MD Level, or an MHF at an equal MD Level. A MEP at a higher MD Level

[11]For this purpose, a pseudo-random value is sufficient, provided that the same value is not produced each time a system is reinitialized.

discards the LTM without further processing. An MP at an equal MD Level directs the LTM to its Bridge's Linktrace Responder. Since, for a given MD Level, different Bridge Ports on a given Bridge can be configured with MHFs, MEPs, both, or neither, an LTM can be both forwarded as ordinary data and processed by the Bridge's Linktrace Responder. An LTM that is originated by an Up MEP is forwarded directly to its Bridge's Linktrace Responder and not through the MEP's Active SAP toward the Frame filtering process.

The Linktrace Responder is responsible for processing and forwarding LTMs, and for replying to them with LTRs. All LTMs are subject to the validation criteria of 20.46.4; invalid or incorrectly addressed LTMs are discarded without further processing. After processing the LTM, the Linktrace Responder can return an LTR through the SAP on which the LTM was received and can forward at most one altered copy of the received LTM through a Linktrace Output Multiplexer (LOM) on a different Bridge Port. It never forwards an LTM past a MEP at the LTM's own MD Level or higher, i.e., a MEP that bounds the LTM's MA. An LTM carries an LTM TTL field (21.8.4) that is decremented each time the LTM passes through a Linktrace Responder. The LTM is not forwarded if this field is 0 or 1 when the LTM is received.

The Linktrace Responder replies to the LTM if either:

a) An ordinary data frame, with the same vlan_identifier as the LTM, a destination_address equal to the Target MAC Address field (21.8.6) of the LTM and received on the same Bridge Port as the LTM, would be forwarded through exactly one other Bridge Port (or to just the Management Port); or
b) The UseFDBonly bit of the Flags field of the LTM is 0, the Target MAC Address field (21.8.6) of the LTM is found in the Bridge's MIP CCM Database (19.3.10), and that entry in the MIP CCM Database identifies a Bridge Port (or the Management Port), other than the one on which the LTM was received; or
c) The MAC address of the MP that received the LTM equals the Target MAC Address field of the LTM;

and:

d) LTM TTL field is non-0 when the LTM is received.

The single output Port identified in the preceding item a) or item b) is called the Egress Port. The Linktrace Responder forwards a copy of the LTM through the LOM on the Egress Port, with a new source_address parameter and the LTM TTL field decremented by 1, only if all of the following conditions are met:

e) An LTR was generated for the reasons specified in the preceding item a) or item b), but not item c);
f) The LTM was received by an MHF, not a MEP;
g) The Egress Port does not have a MEP at or above the LTM's MD Level; and
h) The received LTM TTL field was greater than 1.

NOTE 1—When tracing the path to a known MAC address, all of the Bridges on a shared medium will receive the LTM. Making transmission of the LTR and forwarding of the LTM conditional on the Filtering Database query ensures that at most one of those Bridges will reply. Similarly an LTM can be flooded, through a region of the network that has not yet learned the target MAC address, without provoking a storm of LTRs.

LTMs are forwarded immediately in order to minimize the time required to complete a Linktrace operation. LTRs are enqueued for delivery at a later time. If the LTM was received by a Down MEP or a Down MHF, the LTR includes a Reply Ingress TLV (21.9.8) describing that MP. If not received by a Down MEP, and if the Egress Port has an Up MEP or Up MHF configured for the LTM's MA, the LTR includes a Reply Egress TLV (21.9.9). These TLVs report, to the originating MEP, the MIPs and/or the MEP along the path to the targeted MAC address.

If an LTM is forwarded to a shared medium LAN, and if one or more of the other Bridges attached to that LAN have only two Bridge Ports forwarding data for the LTM's vlan_identifier (including that shared medium), and if the MAC address in the LTM's Target MAC Address field (21.8.6) is not present in those Bridge's Filtering Databases (or if the targeted MAC address is present in more than one Bridge's MIP CCM Database), then more than one of those receiving Bridges can respond to the LTM with an LTR. Also, an anomaly in the operation of Bridge or LAN, e.g., a Filtering Database is recording its information incorrectly, can cause LTR responses along multiple paths.

The Reply TTL field in the LTR would be sufficient by itself for the originating MEP Linktrace Initiator to order the LTRs returned along a single path. The LTMs and LTRs also include three values, the Last Egress Identifier (21.9.7.1), Next Egress Identifier (21.9.7.2), and LTM Egress Identifier TLV (21.8.8), that enable the originating MEP to unambiguously construct the tree of paths taken by the LTM, if LTRs are returned along multiple paths.

NOTE 2—If an MA cannot be configured to transmit CCMs more quickly than Max Age causes MAC addresses to be removed from Bridges' Filtering Databases, then the best practice for the system administrator is to trigger an LTM transmission only after triggering a few LBMs to the target MAC address. This helps to ensure that the target MAC address is known to the intermediate Bridges, so that the LBM will return a result, and so that LTRs will be returned only along a single path.

NOTE 3—If unusual conditions prompt responses along multiple paths, the delay in transmitting the LTR reduces the chance that the originating MEP's receiving capabilities will be overwhelmed by the returned LTRs.

NOTE 4—The forwarded LTM uses, as its source address, the address of the MP through which the LTM is forwarded, not the source address of the original LTM. This means that a Bridge's Linktrace Responder need not be fully synchronized with its Relay Entity, as the latter can change the active topology rapidly without the risk of an adjacent Bridge learning the wrong direction for the source MAC address from an LTM. The LTM TTL field ensures that an LTM that is being processed while the active topology is changing will not be forwarded in a persistent loop.

The MEP Linktrace Initiator ignores any TLVs received in an LTR that do not violate the validation criteria of 20.46, but are not specified in this standard, and it ignores any valid Organization-Specific TLV not known to it. Except as otherwise specified in 20.42.3 and 20.42.4, all TLVs, whether known or ignored by the Linktrace Responder, are copied from the received LTM to the forwarded LTM, and from the received LTM to the LTR.

NOTE 5—Forwarding all unknown TLVs in both the LTM and the LTR enables future revisions of this standard, and designers of Organization-Specific TLVs, to add new capabilities.

20.3.3 Linktrace Reply reception

LTRs are transmitted by a Linktrace Responder in response to a received LTM and returned in a unicast frame to the MEP Linktrace Initiator that originated the LTM. The MEP Linktrace Initiator validates the received LTR, discarding it if invalid, if the destination_address parameter does not match the MAC address of the Bridge Port on which the MEP resides, or if the LTR Transaction Identifier field does not match any of those stored in the MEP's Linktrace database (20.36.2), created when the LTM was originated.

Each validated LTR triggers the creation of a new entry in the Linktrace database, accessible as a managed object through the Read Linktrace Reply command (12.14.7.5). If no entry exists in the Linktrace database for this MEP, corresponding to the LTR Transaction Identifier field, the MEP Linktrace Initiator creates a new entry, with an index value [item c) in 12.14.7.5.2] of 1. If an entry does exist matching that field, it creates a new entry, with an index value one greater than the last matching LTR. In either case, the MEP Linktrace Initiator stores the contents of the LTR in this new entry.

See J.5 for a description of a method to reconstruct the path(s) taken by an LTM, based on the information in the Linktrace database.

20.4 Connectivity Fault Management state machines

The relationships among a MEP's state machines are illustrated in Figure 20-1. That figure uses conventions similar to those of 13.19 and of Figure 13-9: open arrows denote variables that are set by one state machine and both read and set by another; closed arrows denote variables that are set by the owning state machine, and only read by other state machines. Not included in Figure 20-1 are the MEP Continuity Check Initiator state machine, the MP Loopback Responder state machine used in the MEP, and all of the MHF and Linktrace Responder state machines, because they operate independently.

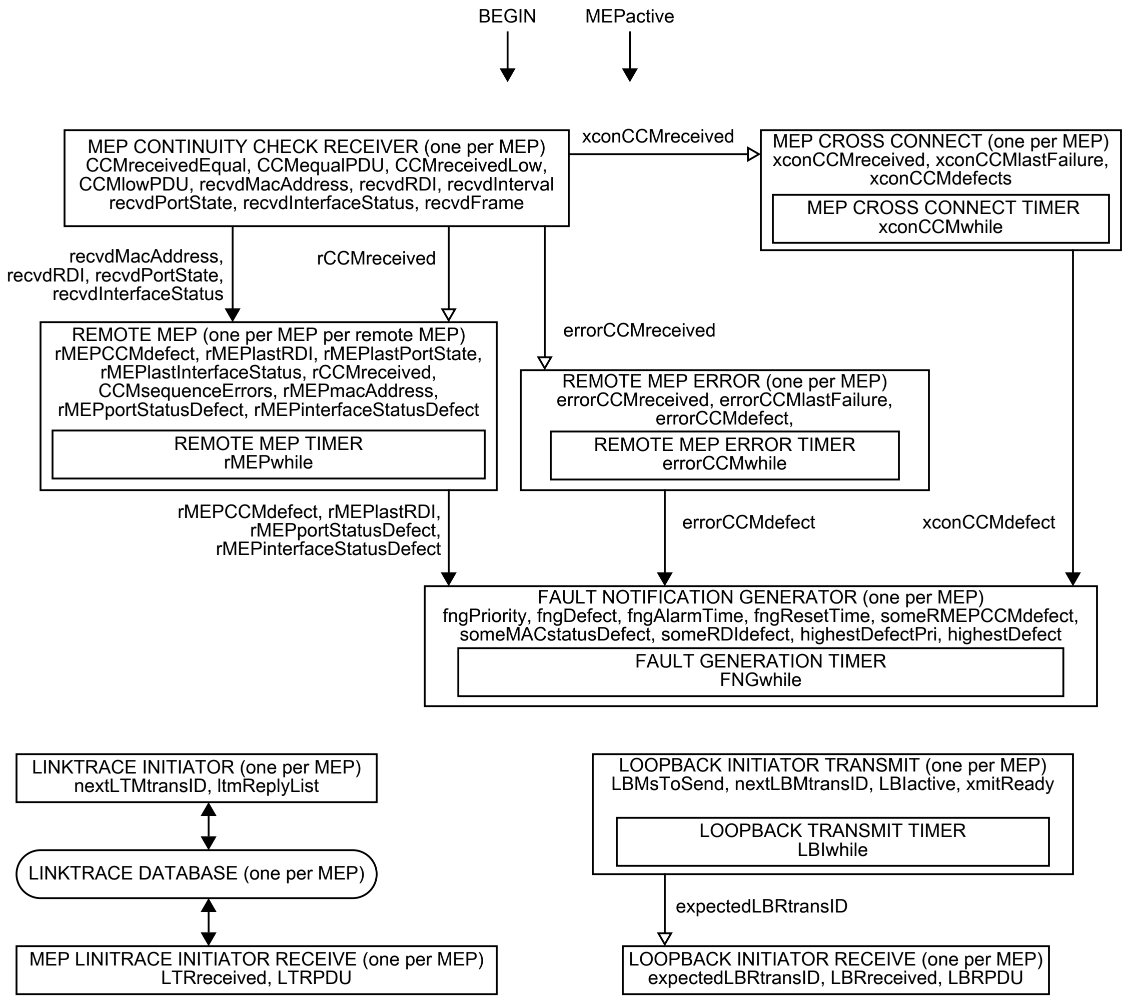
BEGIN
MEPactive
MEP CONTINUITY CHECK RECEIVER (one per MEP)
CCMreceivedEqual, CCMequalPDU, CCMreceivedLow,
CCMlowPDU, recvdMacAddress, recvdRDI, recvdInterval
recvdPortState, recvdInterfaceStatus, recvdFrame
xconCCMreceived
MEP CROSS CONNECT (one per MEP)
xconCCMreceived, xconCCMlastFailure,
xconCCMdefects
MEP CROSS CONNECT TIMER
xconCCMwhile
recvdMacAddress,
recvdRDI, recvdPortState,
recvdInterfaceStatus
rCCMreceived
errorCCMreceived
REMOTE MEP (one per MEP per remote MEP)
rMEPCCMdefect, rMEPlastRDI, rMEPlastPortState,
rMEPlastInterfaceStatus, rCCMreceived,
CCMsequenceErrors, rMEPmacAddress,
rMEPportStatusDefect, rMEPinterfaceStatusDefect
REMOTE MEP TIMER
rMEPwhile
REMOTE MEP ERROR (one per MEP)
errorCCMreceived, errorCCMlastFailure,
errorCCMdefect,
REMOTE MEP ERROR TIMER
errorCCMwhile
rMEPCCMdefect, rMEPlastRDI,
rMEPportStatusDefect,
rMEPinterfaceStatusDefect
errorCCMdefect
xconCCMdefect
FAULT NOTIFICATION GENERATOR (one per MEP)
fngPriority, fngDefect, fngAlarmTime, fngResetTime, someRMEPCCMdefect,
someMACstatusDefect, someRDIdefect, highestDefectPri, highestDefect
FAULT GENERATION TIMER
FNGwhile
LINKTRACE INITIATOR (one per MEP)
nextLTMtransID, ltmReplyList
LINKTRACE DATABASE (one per MEP)
MEP LINITRACE INITIATOR RECEIVE (one per MEP)
LTRreceived, LTRPDU
LOOPBACK INITIATOR TRANSMIT (one per MEP)
LBMsToSend, nextLBMtransID, LBIactive, xmitReady
LOOPBACK TRANSMIT TIMER
LBIwhile
expectedLBRtransID
LOOPBACK INITIATOR RECEIVE (one per MEP)
expectedLBRtransID, LBRreceived, LBRPDU

Figure 20-1—MEP state machines—overview and relationships

20.5 CFM state machine timers

The timer variables declared in this subclause are part of the specification of the operation of CFM. The accompanying descriptions of their meaning and use are provided to aid in the comprehension of the protocol only and are not part of the specification.

One instance of the following shall be implemented:

a) LTFwhile (20.5.1).

One instance of the following shall be implemented per MEP:

b) CCIwhile (20.5.2);
c) errorCCMwhile (20.5.3);
d) xconCCMwhile (20.5.4);
e) LBIwhile (20.5.5); and
f) FNGwhile (20.5.6).

One instance per MEP of the following shall be implemented for each remote MEP in the MEP's MA (those MEPs in the MA other than the MEP, itself):

g) rMEPwhile (20.5.7).

The "granularity" of a timer variable is the periodicity with which it is decremented.

20.5.1 LTFwhile

A timer variable used by the LTR Transmitter state machine to time out the expected transmission of LTRs.

20.5.2 CCIwhile

Timer counter for transmitting CCMs. CCIwhile has a granularity finer than or equal to 1/4 of the time represented by the CCMinterval variable.

20.5.3 errorCCMwhile

Timer counter for timing out invalid CCMs. errorCCMwhile has a granularity finer than or equal to 1 ms.

20.5.4 xconCCMwhile

Timer counter for timing out cross connect CCMs. xconCCMwhile has a granularity finer than or equal to 1 ms.

20.5.5 LBIwhile

A timer variable used by the MEP Loopback Initiator transmit state machine to time out the expected reception of LBRs.

20.5.6 FNGwhile

A timer variable used by the MEP Fault Notification Generator state machine to wait for defects to stabilize and disappear.

20.5.7 rMEPwhile

Timer counter for timing out CCMs. rMEPwhile has a granularity finer than or equal to 1/4 of the time represented by the CCMinterval variable. A Bridge shall not set rMEPCCMdefect within (3.25 $*$ CCMtime(CCMinterval)) seconds of the receipt of a CCM, and shall set rMEPCCMdefect within (3.5 $*$ CCMtime(CCMinterval)) seconds after the receipt of the last CCM. This variable is readable as a managed object [item c) in 12.14.7.6.3].

NOTE—In order to generate a Defect upon the loss of three CCMs, but not generate a Defect when only two CCMs have been lost, a minimum resolution is required for rMEPwhile. The granularity specified for rMEPwhile, when used with the state machine in Figure 20-5, generates a Defect in a maximum of (3.5 ∗ CCMtime(CCMinterval)) seconds, and a minimum of (3.25 ∗ CCMtime(CCMinterval)) seconds.

20.6 CFM procedures

The following procedure is global to the system:

a) CCMtime() (20.6.1).

20.6.1 CCMtime()

CCMtime() takes, as a parameter, a value of the form used in the CCMinterval variable (20.8.1) and the CCM Interval field in a CCM PDU (21.6.1.3). It returns, as a value, the corresponding time interval, in the form used by the CCIwhile or rMEPwhile timer, as appropriate (20.5.2, 20.5.7).

20.7 Maintenance Domain variable

The following variable is local to a single Maintenance Domain and is accessible by more than one state machine:

a) mdLevel (20.7.1).

20.7.1 mdLevel

The integer MD Level of the Maintenance Domain. This variable is available as a managed object [item b) in 12.14.5.1.3].

20.8 Maintenance Association variables

The following variable is local to a single Maintenance Association and is accessible by more than one state machine:

a) CCMinterval (20.8.1).

20.8.1 CCMinterval

The configured time between CCM transmissions. The value configured for this variable shall be one of the non-0 values in Table 21-16. This variable is available as a managed object [item e) in 12.14.6.1.3].

20.9 MEP variables

The following variables are local to a single MEP and are accessible by more than one state machine:

a) MEPactive (20.9.1);
b) enableRmepDefect (20.9.2);
c) MAdefectIndication (20.9.3);
d) allRMEPsDead (20.9.4);
e) lowestAlarmPri (20.9.5);
f) presentRDI (20.9.6); and
g) MEPprimaryVID (20.9.7).

20.9.1 MEPactive

A Boolean indicating the administrative state of the MEP. True indicates that the MEP is to function normally, and false that it is to cease functioning. This variable is available as a managed object [item e) in 12.14.7.1.3].

20.9.2 enableRmepDefect

A Boolean indicating whether frames on the service instance monitored by this MEP's MA are enabled to pass through this Bridge Port by the spanning tree protocol (Clause 13) and VLAN topology management (Clause 11). Set true if either they are enabled, or if the MEP is not associated with a single value of the Port State and VLAN member set, else false. In a Bridge, the value of enableRmepDefect is determined from the Port State (8.4) of the spanning tree instance that controls the MEP's Primary VID, and that VLAN's member set (8.8.9), as shown in Table 20-2. This variable also controls the value output in the Port Status TLV (21.5.4)

Table 20-2—Deriving enableRmepDefect and Port Status TLV in a Bridge

Port State (8.4)	Bridge Port in Primary VID's member set	Port Status TLV (21.5.4)	enableRmepDefect
Disabled, blocked, listening, broken, discarding [a] or learning		psBlocked	False
Forwarding	No	psBlocked	False
	Yes	psUp	True

[a] "Discarding" is used in place of the values disabled, blocked, listening, or broken defined in IETF RFC 4318.

A MEP that is:

a) configured in an MA that is not associated with a VID;
b) on a Bridge that is running multiple instances of the spanning tree;
c) on a Bridge Port that is not excluded by Static VLAN Registration Entries from membership in VIDs belonging to more than one spanning tree instance;

can be unable to generate an unambiguous value for the Port Status TLV, because different MSTIs can be in different Port States. For a MEP in that position, enableRmepDefect is always true.

20.9.3 MAdefectIndication

A Boolean indicating the operational state of the MEP's MA. True indicates that at least one of the remote MEPs configured on this MEP's MA has failed, and false indicates that either all are functioning, or that the MEP has been active for less than the time-out period. MAdefectIndication is true whenever an enabled defect is indicated. That is, MAdefectIndication is true if and only if, for one or more of the variables someRDIdefect, someRMEPCCMdefect, someMACstatusDefect, errorCCMdefect, or xconCCMdefect, that variable is true and the corresponding priority of that variable from Table 20-1 is greater than or equal to the value of the variable lowestAlarmPri.

20.9.4 allRMEPsDead

A Boolean indicating that this MEP is receiving none of the remote MEPs' CCMs. allRMEPsDead is the logical AND of all of the rMEPCCMdefect variables.

20.9.5 lowestAlarmPri

An integer value indicating the lowest defect priority (see Table 20-1) that can trigger the generation of a Fault Alarm. This variable is a managed object [item k) in 12.14.7.1.3].

20.9.6 presentRDI

A Boolean value indicating the state of the RDI bit in CCMs transmitted by this MEP. presentRDI is true if and only if one or more of the variables someRMEPCCMdefect, someMACstatusDefect, errorCCMdefect, or xconCCMdefect is true, and if the corresponding priority of that variable, from Table 20-1, is greater than or equal to the value of the variable lowestAlarmPri.

20.9.7 MEPprimaryVID

An integer specifying a VID, one of the VIDs assigned to the MEP's MA, which is to be used as the Primary VID for this MEP. This variable is related to the managed object item d) in 12.14.7.1.3. It is not necessarily numerically equal to that object, however. MEPprimaryVID always contains the numerical value of the Primary VID. The managed object may contain 0, to indicate that the Primary VID is that of the MEP's MA, or contain the Primary VID itself, if the MA's VID is overridden for this particular MEP.

20.10 MEP Continuity Check Initiator variables

The following variables are local to the MEP Continuity Check Initiator state machine:

a) CCIenabled (20.10.1);
b) CCIsentCCMs (20.10.2); and
c) MACstatusChanged (20.10.3).

20.10.1 CCIenabled

Controls the transmission of CCMs. When set to true, CCMs are transmitted. When set to false, CCM transmission ceases. This variable is available as a managed object [item g) in 12.14.7.1.3].

20.10.2 CCIsentCCMs

Integer value. CCIsentCCMs is initialized to 1 when the MEP is created and may be used thereafter by xmitCCM() to fill the Sequence Number field of a transmitted CCM. This variable is available as a managed object [item w) in 12.14.7.1.3].

20.10.3 MACstatusChanged

Boolean. MACstatusChanged triggers the transmission of one extra CCM. If the Port Status TLV (21.5.4) or Interface Status TLV (21.5.5) is being transmitted by the MEP, and if CCMinterval indicates a transmission interval of 10 s or slower, MACstatusChanged is set to true whenever the value reported in either of those two TLVs changes. MACstatusChanged is reset to false by the MEP Continuity Check Initiator state machine.

20.11 MEP Continuity Check Initiator procedures

The following procedure is local to the MEP Continuity Check Initiator state machine:

a) xmitCCM() (20.11.1).

20.11.1 xmitCCM()

xmitCCM() constructs and transmits a CCM on the Active SAP using an M_UNITDATA.request as follows. xmitCCM():

a) Sets the destination_address parameter to the value from Table 8-9 corresponding to the MEP's MD Level;
b) Sets the source_address parameter to the MAC address of the MEP [item i) in 12.14.7.1.3];
c) Sets the priority parameter according to the MEP's managed objects [item h) in 12.14.7.1.3];
d) Sets the drop_eligible parameter to false;
e) Places the MEP's MD Level (20.7.1) in the MD Level field (21.4.1);
f) Fills the CCM Interval field (21.6.1.3) with the CCM transmission interval (20.8.1, item e) in 12.14.6.1.3);
g) Fills the RDI field (21.6.1.1) with the presentRDI variable (20.9.6);
h) Should copy CCIsentCCMs (20.10.2) to the Sequence Number field of the CCM (21.6.3), else copies 0 into that field;
i) Places the MEP's MAID into the appropriate fields of the CCM [item a) in 12.14.1.2.2, item b) in 12.14.5.3.2, and 21.6.5.1 through 21.6.5.6];
j) Places the MEP's MEPID [item g) in 12.14.6.1.3] into the Maintenance association End Point Identifier field (21.6.4);
k) As controlled by the managed objects item e) in 12.14.3.1.3, item d) in 12.14.5.1.3, and item d) in 12.14.6.1.3, places a Sender ID TLV (21.5.3) in the CCM, identifying the transmitting system;
l) Optionally, places a Port Status TLV (21.5.4) in the CCM, reporting the status of the Bridge Port;
m) Optionally, places an Interface Status TLV (21.5.5) in the CCM, reporting the status of the Bridge Port; and
n) Increments CCIsentCCMs by 1, wrapping around from $2^{32} – 1$ to 0.

20.12 MEP Continuity Check Initiator state machine

Each MEP Continuity Check Initiator (19.2.9) instantiates one MEP Continuity Check Initiator state machine. The MEP Continuity Check Initiator state machine implements the function specified by the state diagram in Figure 20-2, the variable declarations in 20.10 and the procedures in 20.11.

NOTE—Figure 20-2 indicates that the setting of MACstatusChanged causes CCIwhile to be reset and thus changes the phase of the CCM transmission cycle. An alternative formulation of state machine could transmit an extra CCM, but not reset CCIwhile. Whether or not setting MACstatusChanged in a MEP causes CCIwhile to be reset is not specified by this standard.

20.13 MHF Continuity Check Receiver variables

The following variables are defined for the MHF Continuity Check Receiver:

a) MHFrecvdCCM (20.13.1); and
b) MHFCCMPDU (20.13.2).

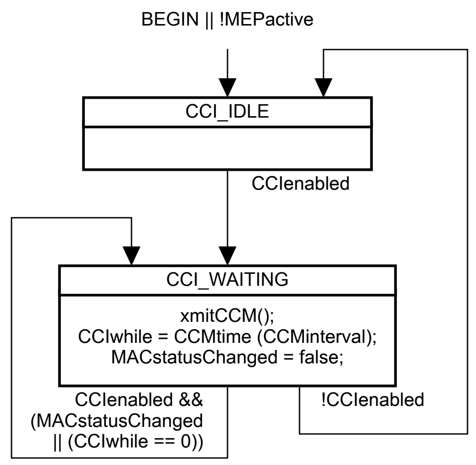
BEGIN || !MEPactive
CCI_IDLE
CCIenabled
CCI_WAITING
xmitCCM();
CCIwhile = CCMtime (CCMinterval);
MACstatusChanged = false;
CCIenabled && (MACstatusChanged || (CCIwhile == 0))
!CCIenabled

Figure 20-2—MEP Continuity Check Initiator state machine

20.13.1 MHFrecvdCCM

A Boolean value set to true by the MHF OpCode Demultiplexer (19.3.6) when a CCM at the MHF's MD Level is received. Cleared by the MHF Continuity Check Receiver state machine.

20.13.2 MHFCCMPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the CCM PDU received by the MHF OpCode Demultiplexer (19.3.6) when a CCM at the MHF's MD Level is received.

20.14 MHF Continuity Check Receiver procedures

The following procedure is defined for the MHF Continuity Check Receiver:

a) MHFprocessCCM() (20.14.1).

20.14.1 MHFprocessCCM()

MHFprocessCCM() shall validate the received CCM according to 20.46.4 and discard the received CCM if invalid. Otherwise, MHFprocessCCM() records the FID, source_address, and Port number of the received CCM in the MHF's MIP CCM Database.

20.15 MHF Continuity Check Receiver state machine

The MHF Continuity Check Receiver state machine implements the functions specified by the state diagram in Figure 20-3, using the variables in 20.13 and procedures in 20.14.

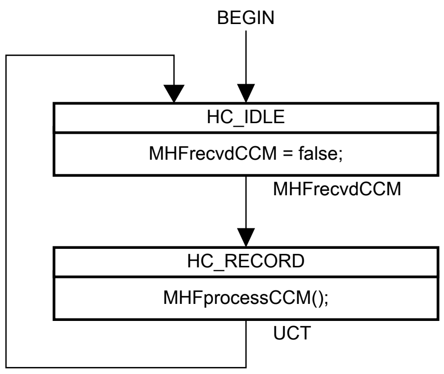

Figure 20-3—MHF Continuity Check Receiver state machine

20.16 MEP Continuity Check Receiver variables

The following variables are defined for the MEP Continuity Check Receiver:

a) CCMreceivedEqual (20.16.1);
b) CCMequalPDU (20.16.2);
c) CCMreceivedLow (20.16.3);
d) CCMlowPDU (20.16.4);
e) recvdMacAddress (20.16.5);
f) recvdRDI (20.16.6);
g) recvdInterval (20.16.7);
h) recvdPortState (20.16.8);
i) recvdInterfaceStatus (20.16.9);
j) recvdSenderId (20.16.10);
k) recvdFrame (20.16.11); and
l) CCMsequenceErrors (20.16.12).

20.16.1 CCMreceivedEqual

Boolean flag. Set by the MEP Equal OpCode Demultiplexer (19.2.7) when a CCM at the MEP's MD Level is received. Cleared by the MEP Continuity Check Receiver state machine.

20.16.2 CCMequalPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the CCM PDU received by the MEP Equal OpCode Demultiplexer (19.2.7) when a CCM at the MEP's MD Level is received.

20.16.3 CCMreceivedLow

Boolean flag. Set by the MEP Low OpCode Demultiplexer (19.2.7) when a CCM below the MEP's MD Level is received. Cleared by the MEP Continuity Check Receiver state machine.

20.16.4 CCMlowPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the CCM PDU received by the MEP Low OpCode Demultiplexer (19.2.7) when a CCM below the MEP's MD Level is received.

20.16.5 recvdMacAddress

Set by the MEPprocessEqualCCM() procedure with the source_address from the last-received CCM from the remote MEP served by this copy of the Remote MEP state machine.

20.16.6 recvdRDI

Boolean flag. Set by MEPprocessEqualCCM() according to the RDI field of the last-received valid CCM.

20.16.7 recvdInterval

Timer counter indicating timer counter equivalent of the value encoded in the CCM Interval field of a received CCM. Set by MEPprocessEqualCCM().

20.16.8 recvdPortState

(optional) Enumerated variable indicating the contents of the Port Status TLV (21.5.4) of the last-received valid CCM, or psNoPortStateTLV: Indicates either that no CCM has been received, or that no Port Status TLV was present in the last CCM received;, if that TLV was not present. Set by MEPprocessEqualCCM().

20.16.9 recvdInterfaceStatus

(optional) Enumerated variable indicating the contents of the Interface Status TLV (21.5.5) of the last-received valid CCM, or isNoInterfaceStatusTLV: Indicates either that no CCM has been received or that no Interface Status TLV was present in the last CCM received; and, if that TLV was not present. Set by MEPprocessEqualCCM().

20.16.10 recvdSenderId

(optional) Enumerated variable indicating the contents of the Sender ID TLV (21.5.3) of the last-received valid CCM, or isNoSenderIdTLV: Indicates either that no CCM has been received, or that no Sender ID TLV was present in the last CCM received., if that TLV was not present. Set by MEPprocessEqualCCM().

20.16.11 recvdFrame

Octet string holding a frame received by MEPprocessEqualCCM(). recvdFrame can be reconstructed either from the M_UNITDATA.indication information presented to the MEP through the Active SAP, or from the EM_UNITDATA.indication presented to the EISS Multiplex Entity. The size of recvdFrame indicates the size of the reconstructed frame. When set to the value NULL, the size of recvdFrame is 0.

20.16.12 CCMsequenceErrors

The total number of out-of-sequence CCMs received from all remote MEPs. This variable is readable as a managed object [item v) in 12.14.7.1.3].

20.17 MEP Continuity Check Receiver procedures

The following procedures are defined for the Continuity Check Receiver:

a) MEPprocessEqualCCM() (20.17.1); and
b) MEPprocessLowCCM() (20.17.2).

20.17.1 MEPprocessEqualCCM()

Called by the MEP Continuity Check Receiver state machine whenever a CCM is received at the MD Level of the MEP. MEPprocessEqualCCM() processes the CCM contained in CCMequalPDU as follows:

a) MEPprocessEqualCCM() shall process the CCM according to 20.46.4.2, and may validate the CCM according to 20.46.4.3, and discard any frames that fail the validation.
b) Otherwise, if the MAID of the received CCM does not exactly match the MAID configured in the receiving ME [(item a) in 12.14.1.2.2, item b) in 12.14.5.3.2] then MEPprocessEqualCCM() sets xconCCMreceived (20.23.1) true, reconstructs the frame containing the CCM into recvdFrame, and places a timer counter value into recvdInterval corresponding to the value of the CCM Interval field in the received CCM.
c) Otherwise, if:
    1) MEPID in the received CCM is not configured in the receiving MEP [item g) in 12.14.6.1.3]; or
    2) MEPID in the received CCM matches the MEPID of the receiving MEP [item b) in 12.14.6.3.2]; or
    3) CCM Interval field in the received CCM does not match that configured for the receiving MEP [item e) in 12.14.6.1.3];

    then MEPprocessEqualCCM() sets errorCCMreceived (20.21.1) true, reconstructs the frame containing the CCM into recvdFrame, and places a timer counter value into recvdInterval corresponding to the value of the CCM Interval field in the received CCM.
d) Otherwise, MEPprocessEqualCCM():
    1) Copies the source_address parameter into recvdMacAddress;
    2) Copies the RDI field to recvdRDI;
    3) Optionally copies the Port Status TLV to recvdPortState;
    4) Optionally copies the Interface Status TLV to recvdInterfaceStatus;
    5) Optionally copies the Sender ID TLV to recvdSenderId;
    6) Optionally updates the Bridge's MIP CCM Database; and
    7) Sets the rCCMreceived variable (20.19.6) for the particular instance of the Remote MEP state machine corresponding to the Maintenance association End Point Identifier field in the CCM.

MEPprocessEqualCCM() should also:

e) Compare the Sequence Number field in the received CCM to the value saved in the MEP CCM Database;
f) If both values are not 0, and if the new value is not 1 greater than the last, increment CCMsequenceErrors (20.16.12); and
g) Store the received Sequence Number in the MEP CCM Database.

20.17.2 MEPprocessLowCCM()

Called by the MEP Continuity Check Receiver state machine whenever a CCM is received with an MD Level that is less than the MEP's MD Level (mdLevel, 20.7.1). MEPprocessLowCCM() processes the CCM contained in CCMlowPDU as follows:

a) MEPprocessLowCCM() shall process the CCM according to 20.46.4.2, and may validate the CCM according to 20.46.4.3, and discard any frames that fail the validation.
b) Otherwise, MEPprocessLowCCM() sets xconCCMreceived (20.23.1) true, reconstructs the frame containing the CCM into recvdFrame, and places a timer counter value into recvdInterval corresponding to the value of the CCM Interval field in the received CCM.

20.18 MEP Continuity Check Receiver state machine

The MEP Continuity Check Receiver state machine implements the functions specified by the state diagram in Figure 20-4, using the variables in 20.16 and procedures in 20.17.

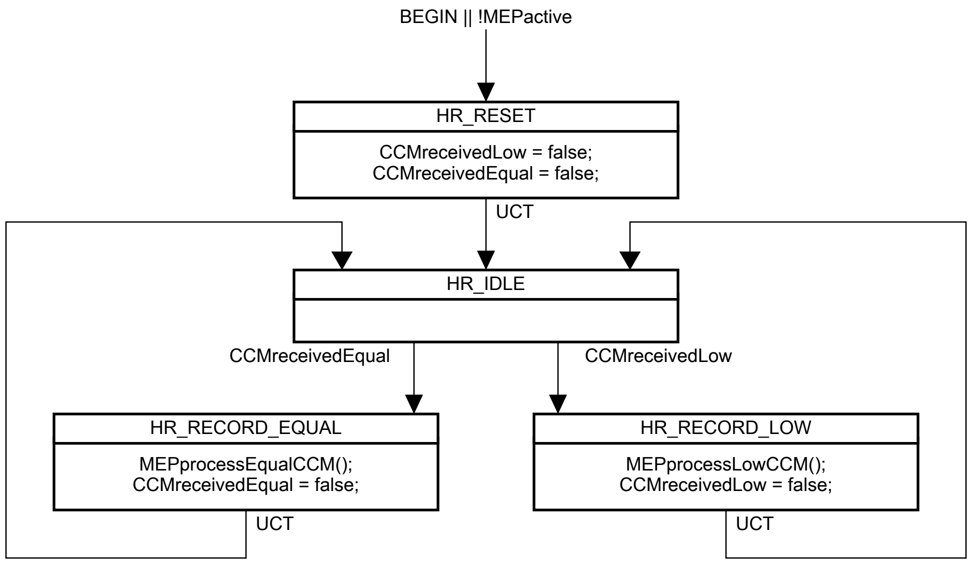

Figure 20-4—MEP Continuity Check Receiver state machine

20.19 Remote MEP variables

The following variables are local to the Remote MEP state machine:

a) rMEPCCMdefect (20.19.1);
b) rMEPlastRDI (20.19.2);
c) rMEPlastPortState (20.19.3);
d) rMEPlastInterfaceStatus (20.19.4);
e) rMEPlastSenderId (20.19.5);
f) rCCMreceived (20.19.6);
g) rMEPmacAddress (20.19.7);
h) rMEPportStatusDefect (20.19.8); and
i) rMEPinterfaceStatusDefect (20.19.9).

20.19.1 rMEPCCMdefect

Reports the state of the remote MEP. When true, no CCM has been received from the remote MEP for at least (3.25 ∗ CCMtime(CCMinterval)) seconds.

20.19.2 rMEPlastRDI

Boolean flag. Contains the RDI flag from the last-received CCM. This variable is readable as a managed object [item e) in 12.14.7.6.3].

20.19.3 rMEPlastPortState

Enumerated value. Contains the value obtained from the Port Status TLV (21.5.4) of the last-received CCM or a value indicating that the last-received CCM contained no Port Status TLV. This variable is readable as a managed object [item f) in 12.14.7.6.3].

20.19.4 rMEPlastInterfaceStatus

Enumerated value. Contains the value obtained from the Interface Status TLV (21.5.5) of the last-received CCM or a value indicating that the last-received CCM contained no Interface Status TLV. This variable is readable as a managed object [item g) in 12.14.7.6.3].

20.19.5 rMEPlastSenderId

Enumerated value. Contains the value obtained from the Sender ID TLV (21.5.3) of the last-received CCM or a value indicating that the last-received CCM contained no Sender ID TLV. This variable is readable as a managed object [item h) in 12.14.7.6.3].

20.19.6 rCCMreceived

Boolean flag set true by MEPprocessEqualCCM() to indicate that a CCM has been received that matches the configured expectations of a particular Remote MEP state machine. Cleared to false by the Remote MEP state machine.

20.19.7 rMEPmacAddress

One field in each entry in the MEP CCM Database for a remote MEP, containing source_address of the last-received CCM from that remote MEP. This variable is readable as a managed object [item d) in 12.14.7.6.3].

20.19.8 rMEPportStatusDefect

A Boolean indicating that the remote MEP is reporting a failure in its Port Status TLV (21.5.4). It is true only if the last received CCM contained a Port Status TLV, and that TLV contained some value other than psUp, i.e., the remote MEP’s Bridge Port is not forwarding data. It is equal to (rMEPlastPortState != psUp && rMEPlastPortState != psNoPortStateTLV: Indicates either that no CCM has been received, or that no Port Status TLV was present in the last CCM received;).

20.19.9 rMEPinterfaceStatusDefect

A Boolean indicating that the remote MEP is reporting a failure in its Interface Status TLV (21.5.5). It is true only if the last received CCM contained an Interface Status TLV that contained some value other than isUp, i.e., the remote MEP’s Bridge Port is not available for forwarding data. It is equal to (rMEPlastInterfaceStatus != isUp && rMEPlastInterfaceStatus != isNoInterfaceStatusTLV: Indicates either that no CCM has been received or that no Interface Status TLV was present in the last CCM received; and).

20.20 Remote MEP state machine

The Remote MEP state machine implements the function specified by the state diagram in Figure 20-5 and the variable declarations in 20.19, and utilizes the procedures in 20.17.

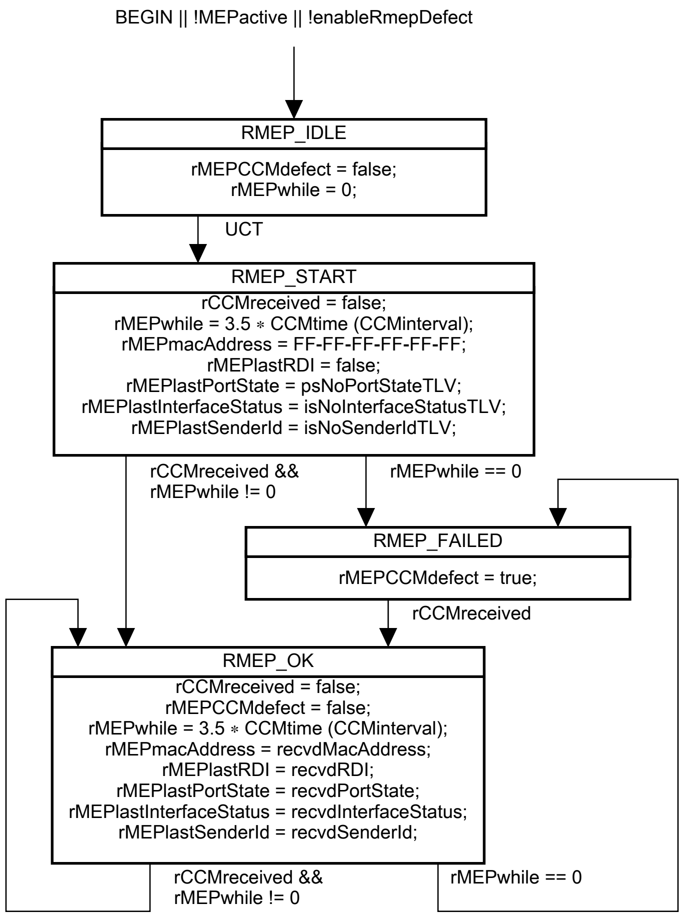
BEGIN || !MEPactive || !enableRmepDefect
RMEP_IDLE
rMEPCCMdefect = false;
rMEPwhile = 0;
UCT
RMEP_START
rCCMreceived = false;
rMEPwhile = 3.5 ∗ CCMtime (CCMinterval);
rMEPmacAddress = FF-FF-FF-FF-FF-FF;
rMEPlastRDI = false;
rMEPlastPortState = psNoPortStateTLV;
rMEPlastInterfaceStatus = isNoInterfaceStatusTLV;
rMEPlastSenderId = isNoSenderIdTLV;
rCCMreceived &&
rMEPwhile != 0
rMEPwhile == 0
RMEP_FAILED
rMEPCCMdefect = true;
rCCMreceived
RMEP_OK
rCCMreceived = false;
rMEPCCMdefect = false;
rMEPwhile = 3.5 ∗ CCMtime (CCMinterval);
rMEPmacAddress = recvdMacAddress;
rMEPlastRDI = recvdRDI;
rMEPlastPortState = recvdPortState;
rMEPlastInterfaceStatus = recvdInterfaceStatus;
rMEPlastSenderId = recvdSenderId;
rCCMreceived &&
rMEPwhile != 0
rMEPwhile == 0

Figure 20-5—Remote MEP state machine

20.21 Remote MEP Error variables

The following variables are local to the Remote MEP Error state machine:

a) errorCCMreceived (20.21.1);
b) errorCCMlastFailure (20.21.2); and
c) errorCCMdefect (20.21.3).

20.21.1 errorCCMreceived

Boolean flag set true by MEPprocessEqualCCM() if an invalid CCM is received. Cleared to false by the Remote MEP Error state machine.

20.21.2 errorCCMlastFailure

Octet string with the same characteristics as recvdFrame. Value controlled by the Remote MEP Error state machine. Readable as a managed object [item t) in 12.14.7.1.3]. Size is sufficient to retain a frame containing a maximum-length CCM in errorCCMlastFailure.

20.21.3 errorCCMdefect

A Boolean flag set and cleared by the Remote MEP Error state machine to indicate that one or more invalid CCMs has been received, and that 3.5 times that CCM's transmission interval has not yet expired. This variable is readable as a managed object [item r) in 12.14.7.1.3].

20.22 Remote MEP Error state machine

The Remote MEP Error state machine implements the function specified by the state diagram in Figure 20-6 and the variable declarations in 20.21.

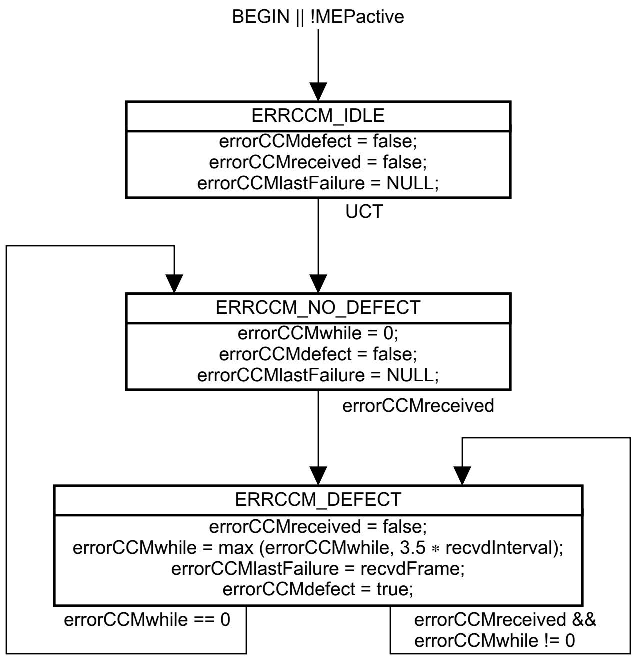
BEGIN || !MEPactive
ERRCCM_IDLE
errorCCMdefect = false;
errorCCMreceived = false;
errorCCMlastFailure = NULL;
UCT
ERRCCM_NO_DEFECT
errorCCMwhile = 0;
errorCCMdefect = false;
errorCCMlastFailure = NULL;
errorCCMreceived
ERRCCM_DEFECT
errorCCMreceived = false;
errorCCMwhile = max (errorCCMwhile, 3.5 * recvdInterval);
errorCCMlastFailure = recvdFrame;
errorCCMdefect = true;
errorCCMwhile == 0
errorCCMreceived && errorCCMwhile != 0

Figure 20-6—Remote MEP Error state machine

20.23 MEP Cross Connect variables

The following variables are local to the MEP Cross Connect state machine:

a) xconCCMreceived (20.23.1);
b) xconCCMlastFailure (20.23.2); and
c) xconCCMdefect (20.23.3).

20.23.1 xconCCMreceived

Boolean flag set true by MEPprocessEqualCCM() when a cross connect CCM is received. Cleared to false by the MEP Cross Connect state machine.

20.23.2 xconCCMlastFailure

Octet string with the same characteristics as recvdFrame. Value controlled by the MEP Cross Connect state machine. Readable as a managed object [item u) in 12.14.7.1.3]. Size is sufficient to retain a frame containing a maximum-length CCM.

20.23.3 xconCCMdefect

A Boolean flag set and cleared by the MEP Cross Connect state machine to indicate that one or more cross connect CCMs has been received, and that 3.5 times of at least one of those CCMs' transmission interval has not yet expired. This variable is readable as a managed object [item s) in 12.14.7.1.3].

20.24 MEP Cross Connect state machine

The MEP Cross Connect state machine implements the function specified by the state diagram in Figure 20-7 and the variable declarations in 20.23.

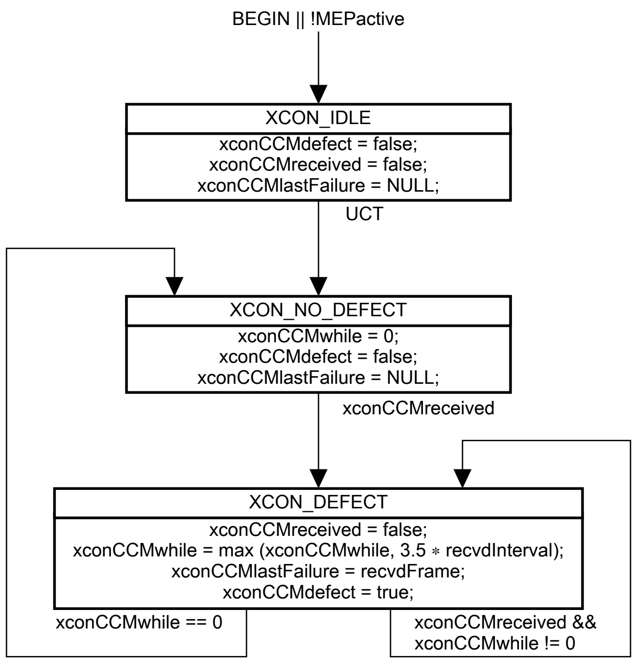

Figure 20-7—MEP Cross Connect state machine

20.25 MP Loopback Responder variables

The following variables are local to the MP Loopback Responder state machine:

a) LBMreceived (20.25.1); and
b) LBMPDU (20.25.2).

20.25.1 LBMreceived

Boolean variable, set to true by an MP OpCode Demultiplexer when an LBM is received, and cleared by an MP Loopback Responder state machine.

20.25.2 LBMPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the LBM PDU received by the an MP OpCode Demultiplexer (19.2.7) when an LBM is received.

20.26 MP Loopback Responder procedures

The following procedures are defined for the Loopback Responder:

a) ProcessLBM() (20.26.1); and
b) xmitLBR() (20.26.2).

20.26.1 ProcessLBM()

Called by the MP Loopback Responder state machine whenever an LBM is received. ProcessLBM() processes the LBM in LBMPDU as follows:

a) If the destination_address parameter contains neither:
    1) the MAC address of the receiving MP; nor
    2) the CCM Group address appropriate to the receiving MP's MD Level, as shown in Table 8-9 (see J.4);

    ProcessLBM() discards the LBM and performs no further processing;
b) If the destination_address parameter contains a Group address and the MP Loopback Responder state machine resides in an MHF (rather than in a MEP), ProcessLBM() discards the LBM and performs no further processing;
c) If the source_address parameter is a Group, and not an Individual MAC address, ProcessLBM() discards the frame and performs no further processing.
d) ProcessLBM() shall process the LBM according to 20.46.4.2, and may validate the LBM according to 20.46.4.3, and discard any frames that fail the validation;
e) If the LBM was not discarded, calls xmitLBR() to generate and transmit an LBR.

20.26.2 xmitLBR()

Called by ProcessLBM() to transmit an LBR. xmitLBR() constructs an LBR from the LBM contained in LBMPDU, and transmits it to the Active SAP using an M_UNITDATA.request as follows. xmitLBR():

a) Sets the destination_address parameter to the source_address of the received LBM;
b) Sets the source_address parameter to the MAC address of the replying MP;
c) Changes the OpCode field (21.4.3) from LBM to LBR;
d) Copies the remainder of the LBM's mac_service_data_unit verbatim to the LBR; and
e) If the replying MP is a MEP, increments the LBR transmission counter by 1 [item ad) in 12.14.7.1.3].

20.27 MP Loopback Responder state machine

The MP Loopback Responder state machine implements the functions specified by the state diagram in Figure 20-8, using the variables in 20.25 and the procedures in 20.26. Although the definition of the MP Loopback Responders in 19.2.10 and 19.3.8 require the instantiation of an MP Loopback Responder state machine in every MP, in practice, the number of MP Loopback Responder state machines implemented in a system cannot be determined from external observation of the system.

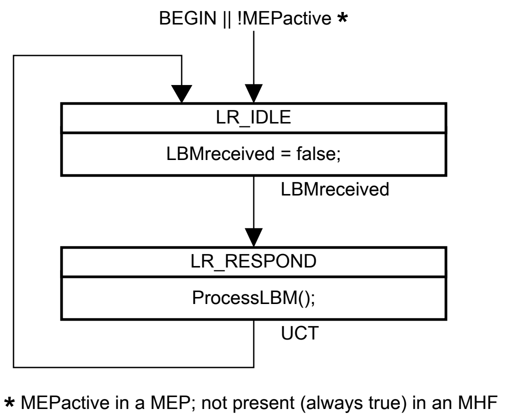

Figure 20-8—MP Loopback Responder state machine

20.28 MEP Loopback Initiator variables

The following variables are local to the MEP Loopback Initiator transmit state machine and the MEP Loopback Initiator receive state machine:

a) LBMsToSend (20.28.1);
b) nextLBMtransID (20.28.2);
c) expectedLBRtransID (20.28.3);
d) LBIactive (20.28.4);
e) xmitReady (20.28.5);
f) LBRreceived (20.28.6); and
g) LBRPDU (20.28.7).

20.28.1 LBMsToSend

The integer number of LBMs that the MEP Loopback Initiator transmit state machine is to transmit. Set by a management operation [item c) in 12.14.7.3.2]. Setting this variable to a non-zero value starts transmission of LBMs. LBMsToSend is decremented by 1 by the MEP Loopback Initiator transmit state machine with each transmission.

20.28.2 nextLBMtransID

The value to place in the Loopback Transaction Identifier field of the next LBM transmitted by xmitLBM(). nextLBMtransID is incremented by 1 by the MEP Loopback Initiator transmit state machine with each transmission. This variable is available as a managed object [item x) in 12.14.7.1.3].

20.28.3 expectedLBRtransID

The value expected to be found in the Loopback Transaction Identifier field of the next LBR received by ProcessLBR(). Altered by ProcessLBR() (see 20.31.1).

20.28.4 LBIactive

A Boolean flag indicating whether the MEP Loopback Initiator transmit state machine is (true) or is not (false) actively engaged in a requested operation. Set to true by the MEP Loopback Initiator transmit state machine after LBMsToSend is set to a non-zero value by a management operation. Reset to false by the MEP Loopback Initiator transmit state machine 5 s after the last LBM is transmitted or the last LBR is received, whichever comes later.

20.28.5 xmitReady

A Boolean flag set to true by the Bridge Port to indicate that another LBM can be transmitted. Reset to false by the MEP Loopback Initiator transmit state machine.

20.28.6 LBRreceived

Boolean flag. Set by the MEP Equal OpCode Demultiplexer (19.2.7) when an LBR at the MEP's MD Level is received. Cleared by the MEP Loopback Initiator receive state machine.

20.28.7 LBRPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the LBR PDU received by the MEP Equal OpCode Demultiplexer (19.2.7) when an LBR at the MEP's MD Level is received.

20.29 MEP Loopback Initiator transmit procedures

The following procedure is local to the MEP Loopback Initiator transmit state machine:

a) xmitLBM() (20.29.1).

20.29.1 xmitLBM()

xmitLBM() is called by the MEP Loopback Initiator transmit state machine. It constructs and transmits an LBM on the Active SAP using an M_UNITDATA.request as follows. xmitLBM():

a) Sets the destination_address parameter from the appropriate managed object [item b) in 12.14.7.3.2];
b) Sets the source_address parameter to the MAC address of the MEP [item i) in 12.14.7.1.3];
c) Sets the priority and drop_eligible parameters from the appropriate managed object [item e) in 12.14.7.3.2];
d) Copies nextLBMtransID (20.28.2) to the Loopback Transaction Identifier field (21.7.3) of the LBM;
e) Constructs a Data TLV from the appropriate managed object [item d) in 12.14.7.3.2] if and only if that managed object has a non-0 length;
f) As controlled by the managed objects item e) in 12.14.3.1.3, item d) in 12.14.5.1.3, and item d) in 12.14.6.1.3, places a Sender ID TLV (21.5.3), identifying the transmitting system, in the LBM; and
g) Increments nextLBMtransID (20.28.2) by 1, wrapping around from $2^{32} – 1$ to 0.

20.30 MEP Loopback Initiator transmit state machine

A MEP creates a single instance of the MEP Loopback Initiator transmit state machine. The MEP Loopback Initiator transmit state machine implements the function specified by the state diagram in Figure 20-9 and the procedures in 20.29. It shares with the MEP Loopback Initiator receive state machine the variable declarations in 20.28.

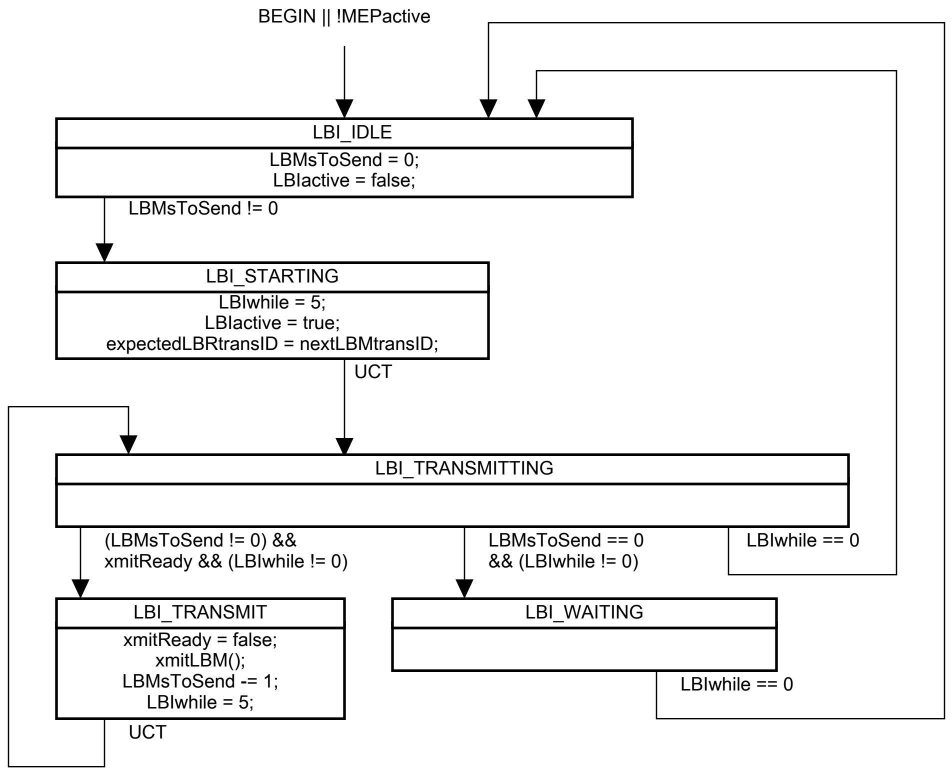

Figure 20-9—MEP Loopback Initiator transmit state machine

20.31 MEP Loopback Initiator receive procedures

The following procedure is local to the MEP Loopback Initiator receive state machine:

a) ProcessLBR() (20.31.1).

20.31.1 ProcessLBR()

Called by the MEP Loopback Initiator receive state machine whenever an LBR is received. ProcessLBR() processes the LBR in LBRPDU as follows:

a) If the I/G bit of the source_address indicates a Group address, or if the destination_address does not match the MAC address of the receiving MP, ProcessLBR() discards the received LBR.
b) ProcessLBR() shall process the LBM according to 20.46.4.2, and may validate the received LBR according to 20.46.4.3, and discard it if invalid.
c) If the LBR is not discarded, and if and only if LBIactive is true, the Loopback Transaction Identifier field of the LBR is compared to expectedLBRtransID:
    1) If the two values are equal, then expectedLBRtransID and the number of correct LBRs received [item y) in 12.14.7.1.3] is incremented by 1; else
    2) The value from the received Loopback Transaction Identifier field is copied into expectedLBRtransID, and the number of incorrect LBRs received [item z) in 12.14.7.1.3] is incremented by 1.

d) ProcessLBR() may perform a bit-by-by comparison of the received LBR against the LBM with the matching Loopback Transaction Identifier, except for the OpCode field, and increment a managed object [item aa) in 12.14.7.1.3] if they do not match.

20.32 MEP Loopback Initiator receive state machine

A MEP creates a single instance of the MEP Loopback Initiator receive state machine. The MEP Loopback Initiator receive state machine implements the function specified by the state diagram in Figure 20-10 and the procedures in 20.31. It shares with the MEP Loopback Initiator transmit state machine the variable declarations in 20.28.

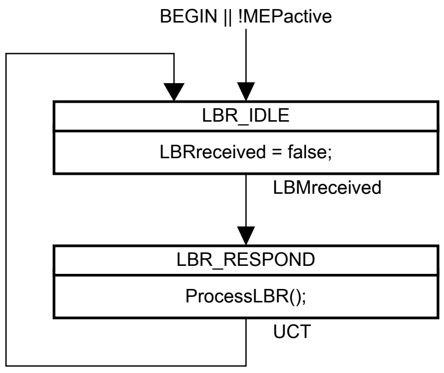

Figure 20-10—MEP Loopback Initiator receive state machine

20.33 MEP Fault Notification Generator variables

The following variables are local to the MEP Fault Notification Generator state machine:

a) fngPriority (20.33.1);
b) fngDefect (20.33.2);
c) fngAlarmTime (20.33.3);
d) fngResetTime (20.33.4);
e) someRMEPCCMdefect (20.33.5);
f) someMACstatusDefect (20.33.6);
g) someRDIdefect (20.33.7);
h) highestDefectPri (20.33.8); and
i) highestDefect (20.33.9).

20.33.1 fngPriority

An integer specifying the priority of the last defect reported in a Fault Alarm. fngPriority takes the same values as highestDefectPri (20.33.8).

20.33.2 fngDefect

An enumerated value specifying the last defect reported in a Fault Alarm. fngDefect takes the same values as highestDefect (20.33.9).

20.33.3 fngAlarmTime

The time that one or more defects must be present before a Fault Alarm is issued. Default value 2.5 s. Also a managed object [item l) in 12.14.7.1.3].

20.33.4 fngResetTime

The time, after a Fault Alarm, that no defects must be present before another Fault Alarm is enabled. Default value 10 s. Also a managed object [item m) in 12.14.7.1.3].

20.33.5 someRMEPCCMdefect

A Boolean indicating the aggregate state of the Remote MEP state machines. True indicates that at least one of the Remote MEP state machines is not receiving valid CCMs from its remote MEP, and false that all Remote MEP state machines are receiving valid CCMs. someRMEPCCMdefect is the logical OR of all of the rMEPCCMdefect variables for all of the Remote MEP state machines on this MEP. This variable is readable as a managed object [item q) in 12.14.7.1.3].

20.33.6 someMACstatusDefect

A Boolean indicating that one or more of the remote MEPs is reporting a failure in its Port Status TLV (21.5.4) or Interface Status TLV (21.5.5). It is true if either some remote MEP is reporting that its interface is not isUp (i.e., at least one remote MEP's interface is unavailable), or if all remote MEPs are reporting a Port Status TLV that contains some value other than psUp (i.e., all remote MEPs' Bridge Ports are not forwarding data). It is thus the logical OR of the following two terms:

a) The logical AND, across all remote MEPs, of the rMEPportStatusDefect variable; OR
b) The logical OR, across all remote MEPs, of the rMEPinterfaceStatusDefect variable.

This variable is readable as a managed object [item p) in 12.14.7.1.3].

20.33.7 someRDIdefect

A Boolean indicating the aggregate health of the remote MEPs. True indicates that at least one of the Remote MEP state machines is receiving valid CCMs from its remote MEP that has the RDI bit set, and false that no Remote MEP state machines are receiving valid CCMs with the RDI bit set. someRDIdefect is the logical OR of all of the rMEPlastRDI variables for all of the Remote MEP state machines on this MEP. This variable is readable as a managed object [item o) in 12.14.7.1.3].

20.33.8 highestDefectPri

An integer value indicating the priority of the defect named in the variable highestDefect. See also Table 20-1.

20.33.9 highestDefect

An enumerated value indicating the highest priority defect among the variables xconCCMdefect (20.23.3), errorCCMdefect (20.21.3), someRMEPCCMdefect (20.33.5), someMACstatusDefect (20.33.6), and someRDIdefect (20.33.7), as limited by lowestAlarmPri (20.9.5). This variable is readable as a managed object [item c) in 12.14.7.7.2]. The variables, their priorities, and the enumerated values of highestDefect are shown in Table 20-1.

20.34 MEP Fault Notification Generator procedures

The following procedure is local to the MEP Fault Notification Generator state machine:

a) xmitFaultAlarm() (20.34.1).

20.34.1 xmitFaultAlarm()

Transmits a Fault Alarm (12.14.7.7). The identity of the MEP and the variable fngDefect (20.33.2), specifying the cause of the Fault Alarm, are transmitted in the Fault Alarm PDU. The format and method of transmission of the Fault Alarm is not specified in this standard.

20.35 MEP Fault Notification Generator state machine

A MEP creates a single instance of the MEP Fault Notification Generator state machine. The MEP Fault Notification Generator state machine implements the function specified by the state diagram in Figure 20-11, the variables in 20.33, and the procedure in 20.34. The current state of the MEP Fault Notification Generator state machine is available in a managed object [item f) in 12.14.7.1.3].

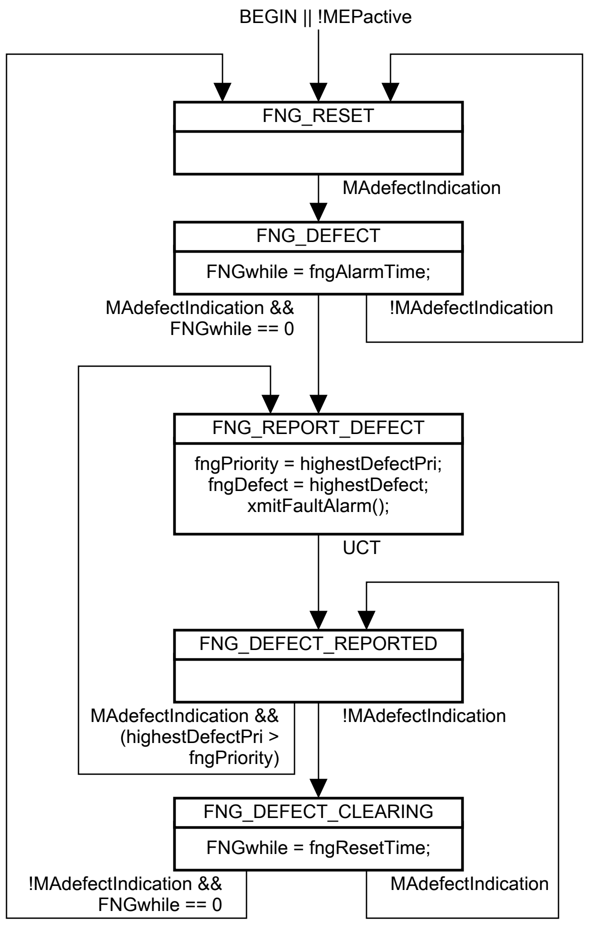

Figure 20-11—MEP Fault Notification Generator state machine

20.36 MEP Linktrace Initiator variables

The following variables are local to the MEP Linktrace Initiator variables:

a) nextLTMtransID (20.36.1); and
b) ltmReplyList (20.36.2).

20.36.1 nextLTMtransID

The value to place in the LTM Transaction Identifier of the next LTM transmitted by xmitLTM(). nextLTMtransID is incremented by 1 by the MEP Linktrace Initiator variables with each transmission. This variable is also a managed object [item ab) in 12.14.7.1.3].

20.36.2 ltmReplyList

The list of recently-issued LTMs and their corresponding LTRs. An LTM entry, with no attached LTR entries, is added to this list by xmitLTM() each time an LTM is transmitted, and an LTR entry is attached to an LTM entry in this list, by ProcessLTR(), each time an LTR corresponding to an LTM in the list is received. The MEP does not remove any entry from this list before 5 s have elapsed since the transmission of the corresponding LTM, unless the addition of another LTM or LTR entry would exceed the maximum resources allocated for this list. This variable constitutes the Linktrace database for a given MEP and is also a managed object (12.14.7.5.3).

Each LTR entry contains the following variables:

a) ltrFlags (20.36.2.1);
b) ltrReplyTTL (20.36.2.2);
c) ltrLastEgressId (20.36.2.3);
d) ltrNextEgressId (20.36.2.4);
e) ltrRelayAction (20.36.2.5);
f) ltrIngressAction (20.36.2.6);
g) ltrIngressAddress (20.36.2.7);
h) ltrIngressPortIdSubtype (20.36.2.8);
i) ltrIngressPortId (20.36.2.9);
j) ltrEgressAction (20.36.2.10);
k) ltrEgressAddress (20.36.2.11);
l) ltrEgressPortIdSubtype (20.36.2.12);
m) ltrEgressPortId (20.36.2.13);
n) ltrSenderIdTlv (20.36.2.14); and
o) ltrOrgSpecTlv (20.36.2.15).

20.36.2.1 ltrFlags

The bit string returned in the Flags field (21.9.1) of the LTR, including the FwdYes and TerminalMEP bits.

20.36.2.2 ltrReplyTTL

The integer value returned in the Reply TTL field (21.9.4) of the LTR.

20.36.2.3 ltrLastEgressId

The octet string returned in the Last Egress Identifier field (21.9.7.1) of the LTR Egress Identifier TLV.

20.36.2.4 ltrNextEgressId

The integer value returned in the Next Egress Identifier field (21.9.7.2) of the LTR Egress Identifier TLV.

20.36.2.5 ltrRelayAction

The enumerated value returned in the Relay Action field (21.9.5) of the LTR. The enumerated values are listed in Table 21-27.

20.36.2.6 ltrIngressAction

The enumerated value returned in the Ingress Action field (21.9.8.1) of the Reply Ingress TLV (21.9.8) of the LTR. The enumerated values are listed in Table 21-30. An enumerated value of 0 indicates that no Reply Ingress TLV was returned in the LTR.

20.36.2.7 ltrIngressAddress

The MAC address of the MP on the Ingress Port. Contents are undefined if no Reply Ingress TLV was returned in the LTR.

20.36.2.8 ltrIngressPortIdSubtype

An enumerated value as specified by IEEE Std 802.1AB™-2005 [B3], 9.5.3.2, indicating the format of ltrIngressPortId. Contents are undefined if no Reply Ingress TLV was returned in the LTR.

20.36.2.9 ltrIngressPortId

An octet string as specified by IEEE Std 802.1AB-2005 [B3], 9.5.9.5, identifying the Ingress Port, in the format identified in ltrIngressPortIdSubtype.

20.36.2.10 ltrEgressAction

The enumerated value returned in the Egress Action field (21.9.9.1) of the Reply Egress TLV (21.9.9) of the LTR. The enumerated values are listed in Table 21-32. A value of 0 indicates that no Reply Egress TLV was returned in the LTR.

20.36.2.11 ltrEgressAddress

The MAC address of the MP on the Egress Port. Contents are undefined if no Reply Egress TLV was returned in the LTR.

20.36.2.12 ltrEgressPortIdSubtype

An enumerated value as specified by IEEE Std 802.1AB-2005 [B3], 9.5.3.2, indicating the format of ltrEgressPortId. Contents are undefined if no Reply Ingress TLV was returned in the LTR.

20.36.2.13 ltrEgressPortId

An octet string as specified by IEEE Std 802.1AB-2005 [B3], 9.5.9.5, identifying the Egress Port, in the format identified in ltrEgressPortIdSubtype.

20.36.2.14 ltrSenderIdTlv

An octet string identifying the transmitting system, as received in the Sender ID TLV (21.5.3), if one was present in the LTR.

20.36.2.15 ltrOrgSpecTlv

An octet string containing the Organization-Specific TLVs (21.5.2), if any were present in the LTR.

20.37 MEP Linktrace Initiator procedures

The following procedure is local to the MEP Linktrace Initiator:

a) xmitLTM() (20.37.1).

20.37.1 xmitLTM()

xmitLTM() is called whenever the Transmit Linktrace Message management operation (12.14.7.4) is invoked. It constructs and transmits an LTM on the Active SAP, or on the MEP LTI SAP if the operation is invoked on an Up MEP, using an M_UNITDATA.request as follows. xmitLTM():

a) Sets the destination_address parameter to the value from Table 8-10 corresponding to the MEP's MD Level;
b) Sets the source_address parameter and Original MAC Address field (21.8.5) to the MAC address of the MEP [item i) in 12.14.7.1.3];
c) Sets the priority parameter to the same value as for CCMs [item h) in 12.14.7.1.3];
d) Copies nextLTMtransID [item ab) in 12.14.7.1.3] to the LTM Transaction Identifier field (21.8.3) of the LTM;
e) Sets the LTM Egress Identifier TLV (21.8.8) to a value that is unique among all Bridges in the Management Domain.
f) Sets the Target MAC Address field (21.8.6) from the appropriate managed object [item c) in 12.14.7.4.2];
g) Sets the LTM TTL field (21.8.4) from the appropriate managed object [item d) in 12.14.7.4.2];
h) Sets the UseFDBonly bit of the Flags field (21.8.1) from the appropriate managed object [item b) in 12.14.7.4.2], and sets all other bits of the Flags field to 0;
i) As controlled by the managed objects item e) in 12.14.3.1.3, item d) in 12.14.5.1.3, and item d) in 12.14.6.1.3, places a Sender ID TLV (21.5.3), identifying the transmitting system, in the LTM;
j) Creates a new entry in the ltmReplyList variable (20.36.2) for this LTM, identified by the LTM Transaction Identifier in nextLTMtransID (20.36.1); and
k) Increments nextLTMtransID (20.36.1) by 1, wrapping around from $2^{32} - 1$ to 0.

If the addition of this LTM entry would exceed the resources allocated to ltmReplyList, then the oldest LTM entries in ltmReplyList are deleted until sufficient resources are available to hold the new LTM entry.

20.38 MEP Linktrace Initiator receive variables

The following variables are local to the MEP Linktrace Initiator receive state machine:

a) LTRreceived (20.38.1); and
b) LTRPDU (20.38.2).

20.38.1 LTRreceived

Boolean flag. Set by the MEP Equal OpCode Demultiplexer (19.2.7) when an LTR at the MEP's MD Level is received. Cleared by the MEP Linktrace Initiator receive state machine.

20.38.2 LTRPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the LTR PDU received by the MEP Equal OpCode Demultiplexer (19.2.7) when an LTR at the MEP's MD Level is received.

20.39 MEP Linktrace Initiator receive procedures

The following procedure is local to the MEP Linktrace Initiator receive state machine:

a) ProcessLTR() (20.39.1).

20.39.1 ProcessLTR()

The ProcessLTR() procedure is called by the MEP Linktrace Initiator receive state machine whenever an LTR is received, and processes the LTR contained in the LTRPDU variable as follows:

a) If the destination_address of the LTR does not match the MAC address of the MEP, or if the LTR fails the validation criteria of 20.46.4, ProcessLTR() shall discard the LTR without counting it, and no further processing takes place.
b) Otherwise, if the LTR Transaction Identifier field matches an LTM entry in the ltmReplyList variable, then a new LTR entry is attached to that LTM entry, containing the information returned in the LTR.
c) Otherwise, the number of unexpected LTRs received [item ac) in 12.14.7.1.3] is incremented by 1.

If the addition of this LTR entry would exceed the resources allocated to ltmReplyList, then the oldest LTM entries in ltmReplyList (the oldest LTR entries, if only one LTM entry is present) are deleted until sufficient resources are available to hold the new LTR entry.

20.40 MEP Linktrace Initiator receive state machine

One instance of the MEP Linktrace Initiator receive state machine is instantiated by each MEP Linktrace Initiator. The MEP Linktrace Initiator receive state machine implements the function specified by the state diagram in Figure 20-12, the variables in 20.38, and the procedures in 20.39.

20.41 Linktrace Responder variables

The following variables are local to the Linktrace Responder:

a) nPendingLTRs (20.41.1);
b) LTMreceived (20.41.2); and
c) LTMPDU (20.41.3).

20.41.1 nPendingLTRs

An integer value used to track the number of LTRs that have been enqueued for transmission by enqueLTR() and not yet transmitted by xmitOldestLTR(). Can be reset to 0 by clearPendingLTRs().

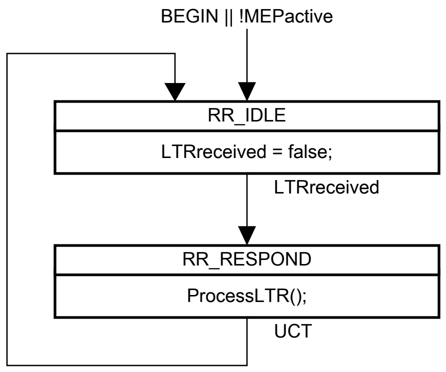

Figure 20-12—MEP Linktrace Initiator receive state machine

20.41.2 LTMreceived

A Boolean value set when a valid LTM is received and cleared by the LTM Receiver state machine.

20.41.3 LTMPDU

Structure. Loaded with the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the LTM PDU received by the MEP Equal OpCode Demultiplexer (19.2.7) when an LTM at the MEP's MD Level is received.

20.42 LTM Receiver procedures

The procedures local to the LTM Receiver state machine are:

a) ProcessLTM() (20.42.1);
b) clearPendingLTRs() (20.42.2);
c) ForwardLTM() (20.42.3); and
d) enqueLTR() (20.42.4).

20.42.1 ProcessLTM()

Called by the LTM Receiver state machine when an LTM is received and processes the LTM contained in the LTMPDU, making the decision whether to call ForwardLTM() to forward the LTM, and whether to call enqueLTR() to enqueue an LTR for transmission, according to 20.3.2. The received LTM is first validated.

a) ProcessLTM() validates the received LTM according to 20.46.3. If the LTM is invalid, no further validation steps are performed.
a) Otherwise, the LTM TTL field (21.8.4) of the received LTM is examined. If its value is 0, the LTM is invalid, and no further validation steps are performed.
b) Otherwise, if the destination_address parameter of the LTM is the Group address appropriate to the MD Level field in the LTM according to Table 8-10, the LTM is valid, and no further validation steps are performed.
c) Otherwise, if both:
    1) The LTM was received from a Linktrace SAP, and not from the originating Up MEP through its MEP LTI SAP; and

2) The destination_address parameter of the LTM is the Individual MAC address of the receiving MP [item i) in 12.14.7.1.3 for a MEP, the MAC address of the Bridge Port for an MHF];

then the LTM is valid.

d) Otherwise, the LTM is invalid.

20.42.1.1 LTM paths through a Bridge

If the LTM is valid, processing proceeds according to the following subclauses. Otherwise, ProcessLTM() discards the LTM, and no further processing takes place.

Figure 20-13 illustrates a number of (but not all) possible paths taken by an LTM through a Bridge. The gray circles indicate the processing of an LTM and the possible forwarding of a modified copy of the LTM. The LTR path is not shown.

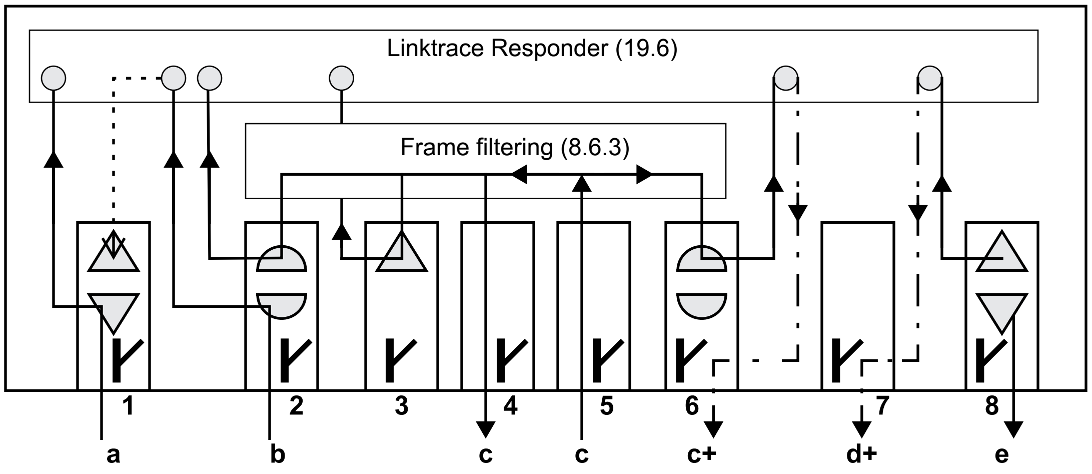

Figure 20-13—Linktrace Responder, MEPs, MHFs, and LOMs

NOTE—All of the MPs shown in Figure 20-13 are at the same MD Level.

a) An LTM is detected by a Down MEP as it enters Port 1. It is deflected to the Linktrace Responder through the MEP Linktrace SAP.
b) An LTM is detected by an MHF as it enters Port 2. It is deflected to the Linktrace Responder through the MHF Linktrace SAP, which determines that the Target MAC Address would be forwarded out Port 1, on which there is a MEP. The LTM is not actually forwarded to the MEP.
c) An LTM enters on Port 5, which has no MHF, and since it is a multicast, the LTM is distributed to Ports 2, 3, 4, and 6 by the Frame filtering function (8.6.3). The original, unaltered LTM passes out Port 4. The Up MEP in Port 3 and the MHFs in Ports 2 and 6 deliver the LTM to the Linktrace Responder. The Linktrace Responder discards the LTMs received on Port 2 and 3, but forwards an updated version of the LTM through the LOM on Port 6.
d) The Up MEP on Port 8 generates an LTM, which it forwards to the Linktrace Responder. The Linktrace Responder decides to forward an updated version of the LTM through the LOM on Port 7. (The Linktrace Responder might equally well have forwarded it through some other Port's MHF.)
e) The Down MEP on Port 8 generates an LTM and transmits it through its Active SAP to the LAN attached to Port 8.

If an LTM is forwarded, the forwarding takes place immediately. If an LTR is generated in response to the LTM, it is enqueued for later transmission by the LTR Transmitter state machine (20.45).

20.42.1.2 Ingress Port, vlan_identifier, and Egress Port determination

In 20.3.2 and Figure 20-13, case a) through case e) all require ProcessLTM() to determine the Ingress and Egress Ports for this received LTM. The Ingress Port is the Bridge Port on which the LTM entered the Bridge, whether through a MEP, an MHF, or an LOM. In case d), where an LTM is generated by an Up MEP and passed through a MEP LTI SAP to the Linktrace Responder, the Ingress Port is the Bridge Port of the originating MEP.

If received from an EISS SAP, the vlan_identifier of the received LTM is included in the EM_UNITDATA.indication. If received from an ISS SAP, the vlan_identifier of the received LTM is that configured in the MEP, MHF, or LOM that received the LTM, or if none, is the PVID (6.7.1) of the Ingress Port.

The Egress Port is the Bridge Port on which a data frame whose destination_address is equal to the Target MAC Address carried in the LTM, and whose vlan_identifier matched that of the LTM, would be forwarded. This determination is made in two steps:

a) ProcessLTM() first queries the Filtering Database (8.8). The set of potential transmission ports, normally created by Active topology enforcement (8.6.1), is the set of all Bridge Ports that are both in the active set of the vlan_identifier of the LTM and that are in the Forwarding state for that vlan_identifier, except that the Ingress Port is excluded from the set. The query uses the Target MAC Address field of the LTM as the destination_address of the lookup, the Original MAC Address field of the LTM as the source_address, and the vlan_identifier of the LTM. The output from this query is a (perhaps reduced) set of potential transmission ports. If the resultant set contains one and only one Bridge Port, that Bridge Port is the Egress Port, and item b) is not performed.

NOTE 1—If there are only two Bridge Ports that are members of the VLAN's member set, the Ingress Port was one of those two Bridge Ports, and the Ingress Port is not connected to a shared medium, then this step can identify the Egress Port even if the Filtering Database is not used for this vlan_identifier.

NOTE 2—This query cannot produce an Egress Port that is the same as the Ingress Port, since the Ingress Port is not included in the set of potential transmission ports.

b) If the Filtering Database could not produce a unique Egress Port, and the MPs serving the vlan_identifier of the LTM are maintaining a MIP CCM Database, and the UseFDBonly bit of the Flags field of the LTM is 0, then ProcessLTM() queries the MIP CCM Database to see whether the target MAC address and vlan_identifier have been retained in that database. If so, and if the Port number in the MIP CCM Database is not the same as the Ingress Port, that Port number identifies the Egress Port.

It is possible that neither of these two steps are able to determine the Egress Port; when a unique Egress Port cannot be determined, an LTM is never forwarded.

20.42.1.3 LTM is received by a Down MEP

In case a) in 20.42.1.1, illustrated in Figure 20-13, the LTM is received by a Down MEP, and delivered to the Linktrace Responder through its MEP Linktrace SAP (19.2.14). In this case, ProcessLTM() performs the following steps:

a) If the Target MAC Address carried in the LTM is the MAC address of the receiving MEP, then the LTM has reached its target. ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MEP Linktrace SAP of the receiving Down MEP.
b) Otherwise, if the spanning tree state of the Bridge Port and vlan_identifier of the LTM is not Forwarding, the LTM is discarded and no further processing takes place.
c) Otherwise, the Ingress and Egress Ports are determined according to 20.42.1.2. If a unique Egress Port cannot be determined, then the LTM is discarded, and no further processing takes place.

d) Otherwise, i.e., a unique Egress Port was found, ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the Linktrace SAP through which the LTM was received. The LTM is discarded, and no further processing takes place.

20.42.1.4 LTM is received by a Down MHF or originated by an Up MEP

In case b) in 20.42.1.1, illustrated in Figure 20-13 on page 173, the LTM is received by a Down MHF. In case d) in 20.42.1.1, illustrated in Figure 20-13, the LTM is originated by an Up MEP. In either case, ProcessLTM() performs the following steps:

a) If the LTM was generated by an Up MEP, item b) and item c) are skipped, and processing continues with item d), as follows.
b) If the Target MAC Address carried in the LTM is the MAC address of the receiving MHF, then the LTM has reached its target. ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the Linktrace SAP through which the LTM was received. The LTM is discarded, and no further processing takes place.
c) Otherwise, if the spanning tree state of the Bridge Port and vlan_identifier of the LTM is not Forwarding, the LTM is discarded and no further processing takes place.

NOTE—This test prevents spurious LTRs from being generated on Backup Ports on a shared medium.

d) Otherwise, the Ingress and Egress Ports are determined according to 20.42.1.2. If a unique Egress Port cannot be determined, then the LTM is discarded, and no further processing takes place.
e) There are three cases for further processing of the LTM, depending on whether, as it passes through the Egress Port, a frame with the LTM's vlan_identifier would first encounter an Up MEP, an Up MHF, or neither. These described in case f), case g), case h), and case i), as follows.
f) If an Up MHF would be encountered on the Egress Port, then:
    1) ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MHF Linktrace SAP of the Egress Port's Up MHF.
    2) If the target MAC address carried in the LTM is the Egress Port Up MHF's MAC address, then the LTM is discarded, and no further processing takes place.
    3) Otherwise, if the LTM TTL field equals 0 or 1, the LTM is discarded and no further processing takes place.
    4) Otherwise, ProcessLTM() calls ForwardLTM() (20.42.3) to forward an altered copy of the LTM through the LOM Linktrace SAP of the LOM on the identified Egress Port.
g) If an Up MEP at the MD Level of the LTM would be encountered on the Egress Port, then:
    1) ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MHF Linktrace SAP of the Ingress Port's Down MHF, or the MEP LTI SAP of the Up MEP, that delivered the LTM to the Linktrace Responder.
    2) The LTM is discarded, and no further processing takes place.
h) Otherwise, if an Up MEP higher than the MD Level of the LTM would be encountered on the Egress Port, then the LTM is discarded, and no further processing takes place.
i) If neither an Up MHF, nor an Up MEP at an MD Level higher than or equal to the LTM, would be encountered on the Egress Port, then:
    1) ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MHF Linktrace SAP of the Ingress Port's Down MHF.
    2) If the LTM TTL field equals 0 or 1, the LTM is discarded and no further processing takes place.
    3) Otherwise, ProcessLTM() calls ForwardLTM() (20.42.3) to forward an altered copy of the LTM through the LOM Linktrace SAP of the LOM that a frame with the LTM's vlan_identifier would first encounter when passing out through the identified Egress Port.

20.42.1.5 LTM is received by an Up MEP

In case c) in 20.42.1.1, illustrated in Figure 20-13, an LTM enters a Bridge on a Bridge Port that has no MEP or MHF, and therefore only an LOM. The LTM can therefore be received by an Up MEP such as the one on Port 3 of Figure 20-13. Processing of an LTM received on an Up MEP proceeds as follows:

a) If the Target MAC Address carried in the LTM is the MAC address of the receiving MEP, then the LTM has reached its target. ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MEP Linktrace SAP of the receiving Up MEP. The LTM is discarded, and no further processing takes place.
b) Otherwise, the Ingress and Egress Ports are determined according to 20.42.1.2. If the Egress Port cannot be determined, or if the Egress Port is not the Bridge Port on which the receiving MEP is configured, then the LTM is discarded and no further processing takes place.
c) Otherwise, (i.e., the Egress Port is that of the receiving Up MEP) ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MEP Linktrace SAP of the receiving Up MEP. The LTM is then discarded.

20.42.1.6 LTM is received by an Up MHF

In case c) in 20.42.1.1, illustrated in Figure 20-13, an LTM enters a Bridge on a Bridge Port that has no MEP or MHF, and therefore only an LOM. The LTM can therefore be received by an Up MHF such as the ones on Port 2 and Port 6 of Figure 20-13. Processing of an LTM received on an Up MHF proceeds as follows:

a) If the Target MAC Address carried in the LTM is the MAC address of the receiving Up MHF (i.e., that of the Bridge Port on which that MHF resides), then the LTM has reached its target. ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MHF Linktrace SAP of the receiving Up MHF. The LTM is discarded, and no further processing takes place.
b) Otherwise, the Ingress and Egress Ports are determined according to 20.42.1.2. If the Egress Port cannot be determined, or if the Egress Port is not the Bridge Port on which the receiving MHF is configured, then the LTM is discarded, and no further processing takes place.
c) Otherwise:
    1) ProcessLTM() calls enqueLTR() (20.42.4) to enqueue an LTR for the MHF Linktrace SAP of the receiving Up MHF.
    2) If the LTM TTL field equals 0 or 1, the LTM is discarded and no further processing takes place.
    3) Otherwise, ProcessLTM() calls ForwardLTM() (20.42.3) to forward an altered copy of the LTM through the MHF Linktrace SAP of the LOM on the same Bridge Port as the Up MHF that received the LTM.

20.42.2 clearPendingLTRs()

Clears the queue of pending LTRs for this MP. Resets nPendingLTRs to 0.

20.42.3 ForwardLTM()

Constructs and transmits a single LTM. Using the original input LTM and the Linktrace Responder SAP through which the LTM (LTR) is to be output, ForwardLTM():

a) Uses the MAC address of the LOM owning the SAP specified for output as the source_address parameter of the LTM;
b) Uses the destination_address and priority parameters of the input LTM as the destination_address and priority of the forwarded LTM;
c) Sets the drop_eligible parameter of the forwarded LTM to false;

d) Uses the same value for the vlan_identifier parameter for the forwarded LTM that was presented with the received LTM;
e) Copies, verbatim, all fields and TLVs, whether known to the Linktrace Responder or not, from the input LTM to the forwarded LTM, except that ForwardLTM():
   1) Places in the forwarded LTM TTL field the value from the input LTM TTL field decremented by 1;
   2) Changes the value in the LTM Egress Identifier TLV to identify the forwarding Linktrace Responder;
   3) Deletes the Sender ID TLV (21.5.3), if present in the LTM; and
   4) Optionally, places a Sender ID TLV (21.5.3), identifying the forwarding system, in the LTM; and
f) Transmits the forwarded LTM on the specified SAP.

20.42.4 enqueLTR()

Constructs and enqueues a single LTR for later transmission by xmitOldestLTR() as follows. Using the input LTM and the Linktrace Responder SAP through which the LTR is to be output, enqueLTR():

a) Uses the MAC address of the MP owning the SAP specified for output as the source_address of the LTR;
b) Uses the MAC address contained in the LTM's Original MAC Address field as the destination_address of the LTR;
c) If the LTR includes a Reply Egress TLV [see the following item p)], uses the Primary VID of the MP on that Egress Port as the vlan_identifier of the LTR, else it uses the Primary VID of the MP on the Ingress Port as the vlan_identifier of the LTR;
d) Sets the priority parameter to the same value as for CCMs [item h) in 12.14.7.1.3];
e) Places its own version in the Version field;
f) If its own version is lower than that of the received LTM, sets all bits and fields in the transmitted PDU that are reserved in its own version, including all bits between the portion of the last header field defined for its own version and the first TLV, to 0;
g) Sets the FwdYes bit of the Flags field to 1 if the LTM was forwarded by the Linktrace Responder, or 0 if not;
h) Sets the TerminalMEP bit of the Flags field to 1 if the MP reported in either the Reply Ingress TLV or the Reply Egress TLV is a MEP, or 0 if not;
i) Copies the Flags field, excepting the FwdYes bit and the TerminalMEP bit, from the LTM to the LTR;
j) Copies the LTM Transaction Identifier field from the LTM to the LTR Transaction Identifier field of the LTR;
k) Copies the LTM Egress Identifier TLV (21.8.8) value from the LTM to the Last Egress Identifier field (21.9.7.1) of the LTR Egress Identifier TLV, or places 0 in that field, if there is no LTM Egress Identifier TLV in the received LTM (see J.4);
l) Sets the Next Egress Identifier field (21.9.7.2) of the LTR Egress Identifier TLV to a value that identifies the forwarding Linktrace Responder (see 21.8.8);
m) Places one less than the value in the LTM TTL field of the LTM in the Reply TTL field of the LTR;
n) Sets the Relay Action field according to Table 21-27;
o) If the LTM was not received by a Down MEP or Down MHF, does not place a Reply Ingress TLV in the LTR; otherwise:
   1) Fills the Ingress Action field of a Reply Ingress TLV (21.9.8) with the appropriate value according to Table 21-30;
   2) Places the receiving MP's MAC address in the Ingress MAC Address field of the Reply Ingress TLV; and
   3) Optionally, fills the remainder of the Reply Ingress TLV with the receiving MP's Port ID information;

p) If the LTM was received by a Down MEP, or if no Egress Port was identified, or if no Up MEP nor Up MHF belonging to the LTM's MA is configured on the Egress Port, does not place a Reply Egress TLV in the LTR; otherwise:
   1) Fills the Egress Action field of a Reply Egress TLV (21.9.9) with the appropriate value according to Table 21-32;
   2) Places the Egress Port's Up MP's MAC address in the Egress MAC Address field of a Reply Egress TLV in the LTM; and
   3) Optionally, fills the remainder of the Reply Egress TLV with Egress Port's Port ID information;

q) As controlled by the managed objects item e) in 12.14.3.1.3, item d) in 12.14.5.1.3, and item d) in 12.14.6.1.3, places a Sender ID TLV (21.5.3), identifying the replying Bridge, in the LTM;

r) Copies, verbatim, all other TLVs in the input LTM to the LTR, except for the Sender ID TLV (21.5.3), which is not copied; and

NOTE 1—Forwarding all unknown TLVs in both the LTM and the LTR enables future revisions of this standard, and designers of Organization-Specific TLVs, to add new capabilities.

NOTE 2—The resultant LTR can be larger than the maximum frame size for the media-dependent sublayer on which the LTR is to be transmitted, causing the LTR to be discarded.

s) Increments nPendingLTRs by 1.

20.43 LTM Receiver state machine

One instance of the LTM Receiver state machine is instantiated by Bridge's Linktrace Responder. The LTM Receiver state machine implements the function specified by the state diagram in Figure 20-14, the variables in 20.41, and the procedures in 20.42.

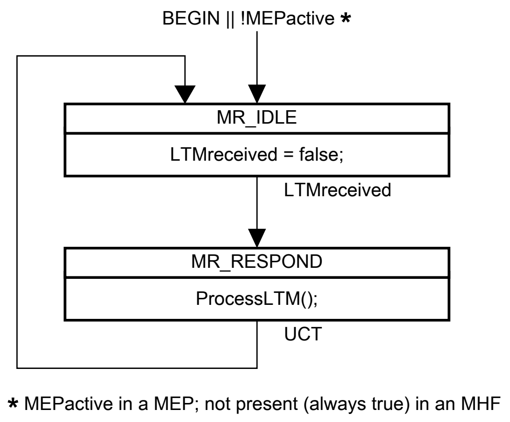

Figure 20-14—LTM Receiver state machine

20.44 LTR Transmitter procedure

The procedure local to the LTR Transmitter state machine is:

a) xmitOldestLTR() (20.44.1).

20.44.1 xmitOldestLTR()

If and only if nPendingLTRs is non-zero, dequeues a single LTR and transmits it, and decrements nPendingLTRs by 1.

20.45 LTR Transmitter state machine

One instance of the LTR Transmitter state machine is instantiated by Bridge's Linktrace Responder. The LTR Transmitter state machine implements the function specified by the state diagram in Figure 20-15, the variables in 20.41, and the procedures in 20.42.

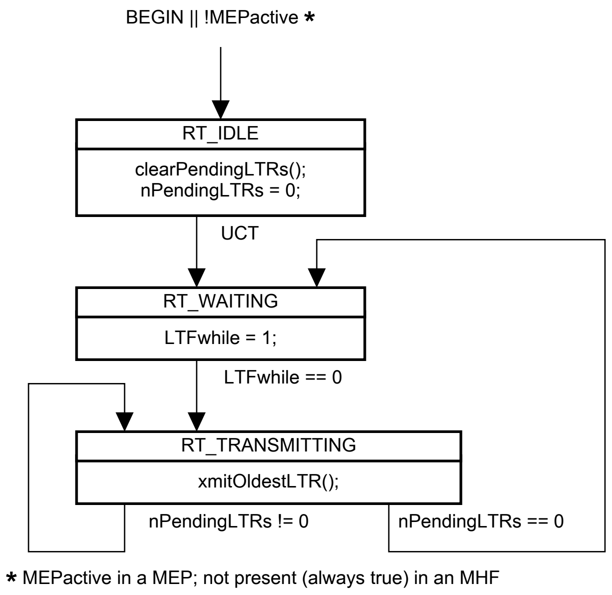

Figure 20-15—LTR Transmitter state machine

This state machine implements the requirement of (20.3.2) to wait a random time interval after receiving an LTM before transmitting an LTR by means of a free running 1 s timer, the expiry of which triggers all LTR transmissions. This method avoids the need to create and manage a timer for every LTM received. Any other implementation that meets the requirements of 20.3.2 can be used in place of this state machine.

20.46 CFM PDU validation and versioning

The purpose of this subclause is to state specific goals to be achieved by CFM with respect to the relationship of this and future versions of this standard, and to define the procedures necessary to meet those goals.

20.46.1 Goals of CFM PDU versioning

The goals of CFM with respect to the relationship of this and future versions of this standard are to ensure that:

a) Implementations of this standard will interoperate with implementations of future versions of this standard;
b) Implementers will be able to offer proprietary, non-standard extensions to this standard with enhanced functionality; and
c) Conformant but extended implementations of this standard will not restrict the ability of future versions of this standard to extend the standard functionality.

20.46.2 PDU transmission

In order to ensure that future versions of Connectivity Fault Management will be compatible with implementations of this standard, certain requirements are placed on transmitted CFM PDUs:

a) The fixed header fields shall be transmitted exactly as specified in this standard.
b) All bits defined as "reserved" in this standard, e.g., unused bits in the Flags field, shall be transmitted as 0.
c) Additional fields shall not be added to the fixed header specified in this standard.
d) Code points reserved in this standard, or in ITU-T Y.1731 (2006), e.g., additional values for the OpCode field in the fixed header (Table 21-4), the Type field in a TLV (Table 21-6), or the reserved values in Table 21-19 for the Maintenance Domain Name Format, shall not be transmitted in any CFM PDU.
e) Additional fields shall not be added to any TLV specified in this standard.

Organization-Specific TLVs can be defined by any organization possessing an OUI, and may be transmitted by an implementation conformant to this standard.

NOTE—The preceding is not an exhaustive list of the restrictions on the transmission of CFM PDUs; it is only a list of the specifications that are most important to future compatibility.

20.46.3 PDU validation

CFM PDUs that fail certain specified tests are discarded by MPs. This requirement is described here as a two-pass processing algorithm, first a Validation Pass, and then an Execution Pass. The Validation Pass is performed first. If and only if it succeeds, the Execution Pass is performed. This description does not preclude other algorithms. For example, an MP could process a CFM PDU in one pass, building a tentative list of changes to the affected state machines and databases, and commit the changes only after the processing has succeeded. However, the behavior of such an implementation shall be indistinguishable, in terms of externally observable behavior (i.e., Management Objects and protocol packets) from the two-pass algorithm described in this subclause.

20.46.4 Validation pass

During the Validation pass, no changes are made to any CFM database other than the updating of certain management counter variables. No protocol state machines are affected, and no changes are made to any CFM PDUs that can be transmitted by the receiving MP. The description of the Validation pass is split into three parts:

a) Operations required of an MP Level Demultiplexer (20.46.4.1);
b) Operations required of all MP components when performing the Validation pass (20.46.4.2); and
c) Operations that are required of some MP components, and optional for others (20.46.4.3).

20.46.4.1 Validation pass operations required of an MP Level Demultiplexer

If the following test fails, the receiving MP Level Demultiplexer (19.2.6) shall consider the CFM PDU invalid and discard it:

a) The length of the mac_service_data_unit is long enough to contain a complete MD Level field.

20.46.4.2 Validation pass operations required of MP components

All bits declared "reserved" in this standard, e.g., unused bits in the Flags field, shall be ignored by the receiving MP.

The receiving MP processes the CFM PDU in accordance with the numerically lower of 1) the Version field in the CFM PDU; and 2) the highest Version number known to the receiving implementation. That is, a Version 1 implementation receiving a Version 0 CFM PDU processes it according to Version 0 of this standard, and it processes a Version 2 CFM PDU according to Version 1. We place the imposition on future Versions of this standard that all earlier Version implementations can process their CFM PDU properly.

As an example of the use of this rule, if a new Flags field bit is defined in Versions 2 and 3, then a Version 3 implementation receiving a Version 1 CFM PDU will ignore that bit, even though the bit is defined in both Version 2 and Version 3, and even though the bit is actually set.

NOTE 1—One effect of this rule is that an implementation conformant to this Version of this standard, Version 0, ignores the Version field in a received CFM PDU.

The following criteria shall not be used by a CFM entity to validate a received CFM PDU:

a) The fixed header can be longer than the length specified by the version of this standard indicated by the CFM PDU's Version field.
b) Bits can be set in reserved bits of the Flags field.
c) A TLV can have a Type field not specified by this standard.
d) A TLV's Length field can be larger than the value (if any) specified in this standard. (This allows TLVs to be extended in future Versions.)
e) Either the First TLV Offset field, or the Length field of the last TLV, in the CFM PDU, can indicate a position for the first (next) TLV that coincides with the end of the mac_service_data_unit containing the CFM PDU. That is, the End TLV can be missing from the CFM PDU if the frame is longer than or equal to the minimum frame size for the media-dependent MAC sublayer.

NOTE 2—These latter criteria cannot be used by a CFM entity to validate a received CFM PDU. This helps to ensure that the CFM PDUs transmitted by future versions of this standard will be considered valid by implementations of the current version (0) of this standard. These criteria can, however, be used by test equipment in order to test the conformance of a system to this standard, e.g., to the rules in 20.46.2.

20.46.4.3 Validation pass operations required of some receiving MP components

The following Validation pass tests are required of some receiving MP components, and are optional for other MP components, as specified in other subclauses. If performed, and if any test fails, the receiving System shall consider the CFM PDU invalid and discard it.

a) The fixed header length, as determined by the First TLV Offset field (21.4.5), is not shorter than the length specified by the version selected according to 20.46.4.2.
b) The fixed-length header does not run over the end of the mac_service_data_unit.
c) A TLV Length field does not run over the end of the mac_service_data_unit.
d) A TLV Length field does not indicate a length that is shorter than the minimum length for that TLV as specified by the Version field of the CFM PDU.
e) Every header field and TLV in the CFM PDU meets the validity criteria specified in the header field and TLV definitions in Clause 21 labeled **Validation Test:**.

20.46.5 Execution pass

The CFM PDU is processed in the Execution Pass. Changes to the CFM databases can be made, the state machines can change states, and the contents of future CFM PDUs can be affected by these changes. The receiving MP:

a) Processes the CFM PDU in accordance with the numerically lower of 1) the Version field in the CFM PDU; and 2) the highest Version number known to the receiving implementation;

b) Processes only those fields in the fixed header portion of the CFM PDU that are defined in the selected Version of the standard, and ignores any extra octets in the fixed header, if the fixed header is longer than the length specified by the selected Version;
c) Ignores any TLV with a Type field not specified by the selected Version;
d) Does not process any part of the CFM PDU following the End TLV (the lack of an End TLV is not an error);
e) Ignores any octets following those specified by the selected Version, if any TLV's Length field is larger than the value (if any) specified the selected Version; and
f) Ignores all bits undefined in this standard, e.g., unused bits in the Flags field.

Unlike the loss of Spanning Tree BPDUs, the loss of CFM PDUs cannot completely disrupt the function of the network. Also, unlike other control protocols such as the IEEE 802.3 Slow Protocols, CFM PDUs can be presented at an arbitrarily high rate. Therefore, a Bridge shall ensure that receiving CFM PDUs on all Bridge Ports at the maximum possible rate supported by the MAC services underlying those ports shall not significantly increase the probability of the failure of the Bridge to maintain loop-free forwarding paths in the Bridged Network. Rather, the Bridge's CPU can be protected against excess CFM PDUs in a similar manner as for other potential Denial of Service attacks.

20.46.6 Future extensions

It is expected that future versions of this protocol will utilize the above versioning rules in order to extend the CFM PDU format in any number of ways, including perhaps:

1) Adding new fields to the header by increasing the minimum size of the First TLV Offset field, and placing the new header fields after the First TLV Offset field in the Version 0 header.
2) Adding new TLVs and extending the lengths of existing TLVs with new fields.
3) Adding a new TLV, that is the first TLV to be processed, by renaming the First TLV Offset field to the "First Version 0 TLV Position," adding a new "Version X Header Length" field following the fields in the Version 0 header, and inserting one or more Version X TLVs just ahead of the first required Version 0 TLV.
4) Adding new information, perhaps fixed-length information, that is required to follow the End TLV specified in Version 0.

Other methods for extending the CFM PDU format are certainly possible. It is therefore critical that Version 0 implementations adhere to the versioning requirements of this clause.

NOTE—The extension options discussed in this Clause are reserved for use by future versions of this standard. Each one is a violation of 20.46.2, 20.46.4, and/or 20.46.5. The Organization-Specific TLV is the only available compliant extension to this standard.

20.47 PDU identification

A received CFM PDU is associated with a particular MP. Every data frame, including one carrying a CFM PDU, has some means, either explicit or implicit, by which that frame can be associated by the receiver with a particular service instance. In a Bridge that is not VLAN-aware, there is only one service instance. In a VLAN-aware Bridge, the vlan_identifier (6.8) identifies the service instance. The means by which the appropriate receiving MP for processing a received CFM PDU is selected are:

1) The direction (PHY side or MAC Relay Entity side) from which the CFM PDU was received;
2) The implicit or explicit association of the frame with a particular service instance (the vlan_identifier); and
3) The MD Level field in the CFM Header.

Once the frame containing the CFM PDU has been delivered to the appropriate entity within an MP, the destination_address can be used to decide whether the CFM PDU is to be processed or discarded, as specified in the description of each entity.

The MAID and/or Maintenance association End Point Identifier field is not used to identify the receiving MP. Thus, if the Individual MP address model (see J.6) is used by a particular Bridge, and each MP uses its Bridge Port's individual MAC address, then the VID, MD Level, and flow direction uniquely identify the receiving MEP. If the Shared MP address model is used by a Bridge, so that Up MPs in the Bridge can share a MAC address, then the specific Bridge Port that is the target of an LBM can be genuinely ambiguous. This ambiguity allows the designer to trade the value of detailed information against cost of obtaining that information.

20.48 Use of transaction IDs and sequence numbers

Each CCM, LBM, and LTM has a field (21.6.3, 21.7.3, and 21.8.3) that is incremented for each PDU transmitted of each type, so that consecutively transmitted PDUs are in numerical order. The variables that control these fields are CCIsentCCMs (20.10.2, for CCM), nextLBMtransID (20.28.2, for LBM), and nextLTMtransID (20.36.1, for LTM).

In theory, every MEP in a Bridge has a CCIsentCCMs variable, a nextLBMtransID variable, and a nextLTMtransID variable, independently from every other MEP. Therefore, any number of MEPs in a single Bridge could be transmitting independent streams of LBMs and LTMs as long as they have different MAC addresses. (If they had the same MAC address, they could not tell each others' LBRs and LTRs apart.)

In practice, there can be fewer instances of these variables and their corresponding state machines than one each per MEP. The managed objects in Clause 12, particularly the definitions of the MEP's LBR counters [item y) in 12.14.7.1.3 and item z) in 12.14.7.1.3] and the responses permitted to the Transmit Loopback Messages command (12.14.7.3.3), are specified so that any number of MEP Loopback Initiator transmit state machines, from one to the number of MEPs, can be implemented. Each state machine can increment its own nextLBMtransID variable to determine the next transmitted Loopback Transaction Identifier field. Similarly, the allowed responses to the Transmit Linktrace Message command (12.14.7.4.3) provide the same latitude for the number of nextLTMtransID variables implemented. The resources for transmitting LTMs or streams of LBMs can then be used serially, rather than simultaneously, by different MEPs in the same Bridge.

On the other hand, every MEP that is incrementing its CCIsentCCMs variable has its own instance of that variable, and thus its own MEP Continuity Check Initiator state machine, because the Remote MEP state machines receiving those CCMs are tracking each MEP's sequence of CCMs independently. A MEP can also transmit 0 in every CCM's Sequence Number field.

Insert a new Clause 21 as follows.

21. Encoding of CFM Protocol Data Units

This clause specifies the method of encoding CFM Protocol Data Units (PDUs). The specifications include the following:

a) Format used to encapsulate or decapsulate a CFM PDU in a frame (21.2);
b) Format of the Common CFM Header, used in all CFM PDUs (21.4);
c) Format used for all Type, Length, Value (TLV) information elements that can be included in CFM PDUs (21.5); and
d) Formats of the Continuity Check Message (CCM, 21.6), the Loopback Message and Loopback Reply (LBM and LBR, 21.7), the Linktrace Message (LTM, 21.8), and Linktrace Reply (LTR, 21.9).

NOTE—Clause 18 introduces the principles of CFM operation and the network architectural concepts that support it. Clause 19 breaks down the CFM protocol entities into their components. Clause 20 specifies the protocols operated by the components of each MP. The use of CFM within systems and networks is further described in Clause 22.

21.1 Structure, representation, and encoding

All CFM PDUs shall contain an integral number of octets.

The octets in a CFM PDU are numbered starting from 1 and increasing in the order they are put into the MAC Service Data Unit (MSDU) that accompanies a request to or indication from the instance of the MAC Internal Sublayer Service (ISS or EISS) used by a CFM entity.

The bits in an octet are numbered from 1 to 8 in order of increasing bit significance, where 1 is the least significant bit in the octet.

Where octets and bits within a CFM PDU are represented using a diagram, octets shown higher on the page than subsequent octets and octets shown to the left of subsequent octets at the same height on the page are lower numbered; bits shown to the left of other bits within the same octet are higher numbered.

Where two or more consecutive octets are represented as hexadecimal values, lower numbered octet(s) are shown to the left and each octet following the first is preceded by a hyphen, e.g., 01-80-C2-00-00-00.

When consecutive octets are used to encode a binary number, the lower octet number has the more significant value. When consecutive bits within an octet are used to encode a binary number, the higher bit number has the most significant value. When bits within consecutive octets are used to encode a binary number, the lower octet number composes the more significant bits of the number. A flag is encoded as a single bit, and is set (True) if the bit takes the value 1, and clear (False) otherwise. The remaining bits within the octet can be used to encode other protocol fields.

21.2 CFM encapsulation

The means for identifying CFM PDUs depend on the medium. For media using a Type/Length field, e.g., IEEE 802.3 media, the identification consists of two octets containing the Type value shown (in hexadecimal notation) in Table 21-1. Media requiring an LLC encapsulation (e.g., IEEE 802.11) use the SNAP encoding shown (in hexadecimal notation) in Table 21-2.

Table 21-1—CFM PDU Encapsulation: Type/Length Media

1	2
89	02

Table 21-2—CFM PDU Encapsulation: LLC Media

1	2	3	4	5	6	7	8
AA	AA	03	00	00	00	89	02

21.3 CFM request and indication parameters

21.3.1 destination_address parameter

There are three classes of CFM PDUs in terms of the CFM entities to which they are addressed:

a) Those addressed to all MEPs in a service instance (CCM);
b) Those addressed to the set of MPs immediately adjacent to the transmitting MP, at a certain MD Level, in a service instance (LTM); and
c) Those addressed to a single, specific MP (LBM, LBR, LTR).

Frames carrying CFM PDUs belonging to two different MAs, but at the same MD Level, are distinguished from each other in the same manner as data frames are distinguished from each other in the corresponding service instances monitored by those MAs.

In a VLAN-aware Bridge, it is the VID that distinguishes service instances. CCMs monitoring a service instance distinguished by its VID use the Group MAC addresses listed in Table 8-9 as the destination_address. LTMs monitoring a VID-distinguished service instance use those shown in Table 8-10. All but the last three bits of the destination_address are fixed by the OpCode, either CCM or LTM. As shown in these two tables, the last three bits of the destination_address match the MD Level field of the CFM PDU Header.

21.3.2 source_address parameter

The Individual MAC address of the MP transmitting the PDU. This is the MAC address of the transmitting MP. An MP’s MAC address is not necessarily unique; MPs configured on the same Bridge Port can share the same MAC address. The principles used to allocate Organization Unique Identifiers (OUIs) required that universally unique MAC addresses (those with the U/L bit = 0, see IEEE Std 802-2001, 9.2) not be used to create MP MAC addresses separately from a physical instance of an IEEE 802 MAC. See J.6 for a discussion of possible assignments of MAC addresses to MPs.

Validation Test: The source_address parameter contains an Individual, and not a Group, MAC address.

21.4 Common CFM Header

The format of a CFM PDU is shown in Table 21-3. A CFM PDU is identified by the encapsulation described in 21.2. Octets 1 through 4, including the MD Level, Version, OpCode, Flags, and First TLV Offset fields, constitute the Common CFM Header.

Table 21-3—Common CFM Header format

	Octet
MD Level	1 (high-order 3 bits)
Version	1 (low-order 5 bits)
OpCode	2
Flags	3
First TLV Offset	4
Varies with value of OpCode	5
End TLV (0)	First TLV Offset + 5

21.4.1 MD Level

(most-significant 3 bits) Integer identifying the Maintenance Domain Level (MD Level) of the packet. Higher numbers correspond to higher Maintenance Associations, those with the greatest physical reach, with the highest values for customers' CFM packets. Lower numbers correspond to lower Maintenance Associations, those with more limited physical reach, with the lowest values for single Bridges or physical links.

21.4.2 Version

(least-significant 5 bits) The protocol version number, always 0. Ignored on receipt.

21.4.3 OpCode

(1 octet) The OpCode field specifies the format and meaning of the remainder of the CFM PDU. Certain values of the OpCode field are reserved for allocation by IEEE 802.1 and/or other organizations. The values for the various CFM PDU types are shown in Table 21-4. When assigning new OpCode values, pairs of CFM PDUs that operate in stimulus-response fashion, such as the LBM/LBR or the LTM/LTR pairs, will use even/odd pairs of values such that the odd (numerically larger) of the two values is the stimulus, and the even (numerically smaller) the response.

Table 21-4—OpCode Field range assignments

CFM PDU or organization	OpCode range
Reserved for IEEE 802.1	0
Continuity Check Message (CCM)	1
Loopback Reply (LBR)	2
Loopback Message (LBM)	3
Linktrace Reply (LTR)	4
Linktrace Message (LTM)	5
Reserved for IEEE 802.1	6 – 31
Defined by ITU-T Y.1731	32 – 63
Reserved for IEEE 802.1.	64 – 255

21.4.4 Flags

(1 octet) The use of the Flags field is defined separately for each OpCode.

21.4.5 First TLV Offset

(1 octet) The offset, starting from the first octet following the First TLV Offset field, up to the first TLV in the CFM PDU. The value of the First TLV Offset field shall be transmitted as indicated in the specification for each of the OpCode field values, as follows (21.6.2, 21.7.2, 21.8.2, and 21.9.2).

21.5 TLV Format

TLV stands for *Type, Length, Value* and denotes a method of encoding variable-length and/or optional information in a PDU. TLVs are not aligned to any particular word or octet boundary. TLVs follow each other with no padding between TLVs.

21.5.1 General format for CFM TLVs

The TLV format is shown in Table 21-5.

Table 21-5—TLV format

	Octet
Type	1
Length	2 – 3
(Value)	4

21.5.1.1 Type

(1 octet) Required. If 0, no Length or Value fields follow. If not 0, at least the Length field follows the Type field. The Type field is encoded as shown in Table 21-6.

21.5.1.2 Length

(2 octets) Required if the Type field is not 0. Not present if the Type field is 0. The 16 bits of the Length field indicate the size, in octets, of the Value field. 0 in the Length field indicates that there is no Value field.

21.5.1.3 Value

(Length specified by the Length field) Optional. Not present if the Type field is 0 or if Length field is 0.

21.5.2 Organization-Specific TLV

Type = 31. Any organization can define TLVs for use in Connectivity Fault Management. The format for such TLVs is shown in Table 21-7. The "OUI" is an Organizationally Unique Identifier, obtainable from IEEE.[12] The Subtype is required, so that an additional OUI will not be required if more Organization-Specific TLV are required by an owner of an OUI.

[12]Interested applicants should contact the IEEE Standards Department, Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA, http://standards.ieee.org/regauth/index.html.

Table 21-6—Type Field values

TLV or organization	Type field
End TLV	0
Sender ID TLV	1
Port Status TLV	2
Data TLV	3
Interface Status TLV	4
Reply Ingress TLV	5
Reply Egress TLV	6
LTM Egress Identifier TLV	7
LTR Egress Identifier TLV	8
Reserved for IEEE 802.1	9 – 30
Organization-Specific TLV	31
Defined by ITU-T Y.1731	32 – 63
Reserved for IEEE 802.1	64 – 255

Table 21-7—Organization-Specific TLV format

	Octet
Type = 31	1
Length	2 – 3
OUI	4 – 6
Sub-Type	7
Value (optional)	8 – (Length + 3)

This TLV category is provided to allow different organizations, such as IEEE 802.1, IEEE 802.3, ITU-T, as well as individual software and equipment vendors, to define TLVs that advertise information to remote entities attached to the same media, subject to the following restrictions:

1) Information transmitted in an Organization-Specific TLV shall not require the recipient to violate any requirement in this standard.
2) Information transmitted in one Organization-Specific TLV shall not be used to provide a means for sending messages that are larger than would fit within a single CFM PDU.
3) Organization-Specific TLVs shall conform to 20.46.

21.5.3 Sender ID TLV

Type = 1. Identifies the Bridge on which the transmitting MP is configured and may also include a management address for that Bridge. The allowed values are those contained in the Chassis ID TLV and Management Address TLV in 9.5.2 and 9.5.9 of IEEE 802.1AB LLDP, modified to fit into a CFM TLV, as shown in Table 21-8. Whether the Sender ID TLV is transmitted, and what information is included in the TLV, are controlled by managed objects [item e) in 12.14.3.1.3, item d) in 12.14.5.1.3, item d) in 12.14.6.1.3].

Validation Test: The Length field is large enough to contain all of the fields indicated as being present by the Chassis ID Length, Management Address Domain Length, and/or Management Address Length fields.

Table 21-8—Sender ID TLV format

	Octet
Type = 1	1
Length	2 – 3
Chassis ID Length	4
Chassis ID Subtype	5
Chassis ID	6 – (Chassis ID Length + 5)
Management Address Domain Length	(Chassis ID Length + 6)
Management Address Domain	(Chassis ID Length + 7) – (Chassis ID Length + Management Address Domain Length + 6)
Management Address Length	(Chassis ID Length + Management Address Domain Length + 7)
Management Address	(Chassis ID Length + Management Address Domain Length + 8) – (Length + 3)

21.5.3.1 Chassis ID Length

(1 octet) The length, in octets, of the Chassis ID field. 0 if no Chassis ID is specified. Always present.

Validation Test: The Chassis ID Length field is either 0, or is less than (TLV Length field value – 1).

21.5.3.2 Chassis ID Subtype

(1 octet) Identifies the format of the Chassis ID field. Specified by IEEE Std 802.1AB-2005, 9.5.2.2. Not present if the Chassis ID Length field contains 0.

21.5.3.3 Chassis ID

(Length specified by the Chassis ID Length field) Identifies the chassis. Specified by IEEE Std 802.1AB-2005 [B3], 9.5.2.3. Not present if the Chassis ID Length field contains 0.

21.5.3.4 Management Address Domain Length

(1 octet) The Management Address Domain Length field contains the length, in octets, of the Management Address Domain field. If 0, or if the TLV's Length field indicates that the Management Address Domain Length field is not present, then the Management Address Domain, Management Address Length, and Management Address fields are not present.

21.5.3.5 Management Address Domain

(Length specified by the Management Address Domain Length field) The Management Address Domain field specifies the format and type of the contents of the Management Address field, as well as a management mechanism. The format of the Management Address Domain field shall be that of an Object Identifier (OID), as specified by ITU-T X.690 (2002), 8.19. The OID references a TDomain (IETF RFC 2579). Standard values that are useful for the Management Address Domain field include, but are not limited to:

— transportDomainUdpIpv4, indicating SNMP over UDP over IPv4 (IETF RFC 3419);
— transportDomainUdpIpv6, indicating SNMP over UDP over IPv6 (IETF RFC 3419); and

— snmpIeee802Domain, indicating SNMP over the MAC service (IETF RFC 4789).

This field is not present if the Management Address Domain Length field is not present or contains a 0.

21.5.3.6 Management Address Length

(1 octet) The length, in octets, of the Management Address field. This field is not present if the Management Address Domain Length field is not present or contains a 0.

21.5.3.7 Management Address

(Length specified by the Management Address Length field) Identifies a TransportDomain (IETF RFC 3419, IETF RFC 4789) through which the Bridge in which the transmitting MP is configured can be managed. The format of the Management Address field is specified by the MIB module defining the TransportDomain. This field is not present if the Management Address Domain Length field is not present or contains a 0, or if the Management Address Length field is not present or contains a 0.

NOTE—The contents of the Management Address Domain and Management Address are defined in terms of MIB modules. Therefore, a published MIB module is required in order to define a specific valued that can be used in the Management Address Domain field. However, the use of Management Address Domain field by a system does not require that the system be manageable via SNMP; the published MIB module could define, for example, a TransportDomain for a proprietary Command Line Interpreter over ITU-T X.25 (1996) [B15].

21.5.4 Port Status TLV

Type = 2. The Port Status TLV indicates the ability of the Bridge Port on which the transmitting MEP resides to pass ordinary data, regardless of the status of the MAC. The value of this TLV is driven by the MEP variable enableRmepDefect (20.9.2), as shown in Table 21-10. The format of this TLV is shown in Table 21-9. Any change in the Port Status TLV's value triggers one extra transmission of that Bridge Port's MEPs' CCMs.

Table 21-9—Port Status TLV format

	Octet
Type = 2	1
Length	2 – 3
See Table 21-10	4

Table 21-10—Port Status TLV values

mnemonic	Ordinary data passing freely through the Port	value
psBlocked	No: enableRmepDefect = false	1
psUp	Yes: enableRmepDefect = true	2

Validation Test: The Port Status TLV contains one of the values listed in Table 21-10.

A MEP that is configured in a Bridge in a position that is not associated with a single value for the Port State and VID member set, e.g., a MEP in position 5 in Figure 22-9, on a Bridge running multiple spanning tree instances via MSTP, shall not transmit the Port Status TLV.

21.5.5 Interface Status TLV

Type = 4. The Interface Status TLV indicates the status of the interface on which the MEP transmitting the CCM is configured (which is not necessarily the interface on which it resides, see J.6), or the next lower interface in the IETF RFC 2863 IF-MIB. The format of this TLV is shown in Table 21-11. The enumerated values are shown in Table 21-12. These values correspond to the values for ifOperStatus in IETF RFC 2863.

Table 21-11—Interface Status TLV format

	Octet
Type = 4	1
Length	2 – 3
See Table 21-12	4

Table 21-12— Interface Status TLV values

mnemonic	Interface Status (IETF RFC 2863 ifOperStatus)	value
isUp	up	1
isDown	down	2
isTesting	testing	3
isUnknown	unknown	4
isDormant	dormant	5
isNotPresent	notPresent	6
isLowerLayerDown	lowerLayerDown	7

Validation Test: The Interface Status TLV field contains one of the values listed in Table 21-12.

21.5.6 Data TLV

Type = 3. Zero or more octets of arbitrary data. Serves several purposes, including the transmission of different frame sizes to test MTU capabilities and the testing for data-specific error dependencies. The Data TLV may be included in the LBM and the LBR, but not in any other CFM PDU. The contents of the Data TLV shall not be examined or interpreted by the receiver of any CFM PDU except an LBR. The format of the Data TLV is shown in Table 21-13.

Table 21-13—Data TLV format

	Octet
Type = 3	1
Length	2 – 3
Data	4 – (Length + 3)

21.5.7 End TLV

Required. Type = 0. Length and Value fields are not present. The End TLV is the last TLV in the CFM PDU. (Lack of the End TLV does not invalidate a received CFM PDU, but certain cases where frames receive extra headers, e.g., MACsec or additional VLAN tags, can cause errors if it is not present.) The format of the End TLV is shown in Table 21-14.

Table 21-14—End TLV format

	Octet
Type = 0	1

21.6 Continuity Check Message format

The format of the CCM is shown in Table 21-15. In order to limit the resources required to generate and receive CCMs, the following restrictions apply to the format of the CCM:

Table 21-15—Continuity Check Message format

	Octet
Common CFM Header	1 – 4
Sequence Number	5 – 8
Maintenance association End Point Identifier	9 – 10
Maintenance Association Identifier (MAID)	11 – 58
Defined by ITU-T Y.1731	59 – 74
Reserved for definition in future versions of the protocol [a]	
Optional CCM TLVs	First TLV Offset + 5 [b]
End TLV (0)	First TLV Offset + 5, if no Optional CCM TLVs are present

[a] This field has 0 length in this version 0 of CFM. It is shown in order to stress that additional information can be present in future versions of CFM, and that a version 0 receiver ignores its contents, if present.
[b] Octet 75 for transmitted CCMs.

a) The Maintenance Domain Name and Short MA Name are encoded in a manner such that an MP can treat all of the fields from the Maintenance Domain Name Format field through the Short MA Name field as a unit when deciding whether a received CCM does or does not belong to the receiving MP's MA. That is, the MP can perform a binary comparison of these fields in a received CCM to the fields it would transmit in its own CCMs. This means, for example, that two Maintenance Domain Names differing, e.g., only by trailing spaces or by upper/lower case substitutions, can be interpreted to denote different MAs.
b) The portion of the PDU allocated for the MAID is limited to 48 octets, in order to minimize the memory requirements of a hardware implementation of a MEP.

An MP shall be able to receive and process any valid CCM PDU that is 128 octets in length or less, starting with the MD Level/Version octet and including the End TLV. An MP shall not transmit a CCM PDU exceeding this length. An MP may discard as invalid any received CCM PDU that exceeds this length.

21.6.1 Flags

(1 octet) The Flags field of the Common CFM Header is split into three parts for the CCM as follows:

a) RDI field (21.6.1.1);
b) Reserved field (21.6.1.2); and
c) CCM Interval field (21.6.1.3).

21.6.1.1 RDI

The most significant bit of the Flags field is the RDI bit. This bit is set to 1 if the transmitting MEP's presentRDI variable (20.9.6) is set, and 0 if not.

21.6.1.2 Reserved

The bits of the Flags field not including the RDI field and the CCM Interval field are set to 0 by the transmitting MP, and are not be examined by the receiving MP [item b) in 20.46.2].

21.6.1.3 CCM Interval

The least-significant three bits of the Flags field constitute the CCM Interval field. The CCM Interval field is encoded as specified in Table 21-16.

Table 21-16—CCM Interval field encoding

Transmission Interval	max. CCM Lifetime	min. CCM Lifetime	CCM Interval field
invalid			0
3 1/3 ms (300 Hz)	11 2/3 ms	10 5/6 ms	1
10 ms	35 ms	32.5 ms	2
100 ms	350 ms	325 ms	3
1 s	3.5 s	3.25 s	4
10 s	35 s	32.5 s	5
1 min	3.5 min	3.25 min	6
10 min	35 min	32.5 min	7

NOTE—The maximum CCM Lifetime in Table 21-16 is equal to 3.5 times the CCM Interval. The minimum CCM Lifetime is the maximum CCM Lifetime minus 1/4 of the CCM Interval, the minimum required granularity of the timer variables CCIwhile, rMEPwhile, errorCCMwhile, and xconCCMwhile.

Validation Test: The CCM Interval field does not contain the value 0.

21.6.2 First TLV Offset

(1 octet) The First TLV Offset field of the Common CFM Header in a CCM is transmitted as 70.

Validation Test: The First TLV Offset field of the Common CFM Header in a CCM contains a value greater than or equal to 70.

21.6.3 Sequence Number

(4 octets) A MEP transmits either a 0 in this field or copies to it the contents of the CCIsentCCMs variable (20.10.2).

21.6.4 Maintenance association End Point Identifier

(2 octets) Contains an integer value. This TLV specifies from which MEP the CCM was transmitted.

Validation Test: MEPID is in the range 1–8191.

21.6.5 Maintenance Association Identifier

(48 octets) This field contains the MAID of the transmitting MEP. It is divided into six subfields, plus, if necessary, a pad of octets contains 0 to fill out its fixed length. The subfields are:

a) Maintenance Domain Name Format (21.6.5.1);
b) Maintenance Domain Name Length (21.6.5.2);
c) Maintenance Domain Name (21.6.5.3);
d) Short MA Name Format (21.6.5.4);
e) Short MA Name Length (21.6.5.5); and
f) Short MA Name (21.6.5.5).

The total length of these fields, including padding, if present, shall be exactly 48 octets. There are two possible formats of the Maintenance Association Identifier field, as shown in Table 21-17 and Table 21-18, depending on whether or not the Maintenance Domain Name is present.

Table 21-17—CCM MAID field format: Maintenance Domain present

	Octet
Maintenance Domain Name Format (not 1)	1
Maintenance Domain Name Length	2
Maintenance Domain Name	3 – (mdnl [a] + 2)
Short MA Name Format	(mdnl + 3)
Short MA Name Length	(mdnl + 4)
Short MA Name	(mdnl + 5) – (mdnl + smanl [b] + 4)
0 pad, if necessary	(mdnl + smanl + 5) – 48

[a] Maintenance Domain Name Length.
[b] Short MA Name Length.

21.6.5.1 Maintenance Domain Name Format

(1 octet) Specifies the format of the Maintenance Domain Name field. The Maintenance Domain Name Format is encoded according to Table 21-19.

Table 21-18—CCM MAID field format: Maintenance Domain not present

	Octet
Maintenance Domain Name Format (1)	1
Short MA Name Format	2
Short MA Name Length	3
Short MA Name	4 – (smanl [a] + 3)
0 pad, if necessary	(smanl + 4) – 48

[a] Short MA Name Length

Table 21-19—Maintenance Domain Name Format

Maintenance Domain Name Format field	Value
Reserved for IEEE 802.1	0
No Maintenance Domain Name present	1
Domain Name based string [a]	2
MAC address + 2-octet integer	3
Character string [b]	4
Reserved for IEEE 802.1	5 – 31
Defined by ITU-T Y.1731	32 – 63
Reserved for IEEE 802.1	64 – 255

[a] This is a string the terminal substring of which is an RFC1035 DNS Name, and the remainder of which is a text string, following the syntax of a DNS Name, denoting the identity of a particular Maintenance Domain.
[b] This is an IETF RFC 2579 DisplayString, with the exception that character codes 0-31 (decimal) are not used.

21.6.5.2 Maintenance Domain Name Length

(1 octet) Specifies the length in octets of the Maintenance Domain Name field. Not present if the Maintenance Domain Name Format field specifies “No Maintenance Domain Name present” (1).

Validation Test: The Maintenance Domain Name Length in a CCM, if present, is greater than or equal to 1, and less than or equal to 43.

21.6.5.3 Maintenance Domain Name

(Length specified by the Maintenance Domain Name Length field) Contains the Maintenance Domain Name, in the format specified by the Maintenance Domain Name Format field. Not present if the Maintenance Domain Name Format field specifies “No Maintenance Domain Name present” (1).

21.6.5.4 Short MA Name Format

(1 octet) Specifies the format of the Short MA Name field. The Short MA Name Format is encoded according to Table 21-20.

Table 21-20—Short MA Name Format

Short MA Name Format Field	Value
Reserved for IEEE 802.1	0
Primary VID	1
Character string [a]	2
2-octet integer	3
RFC 2685 VPN ID	4
Reserved for IEEE 802.1	5 – 31
Defined by ITU-T Y.1731	32 – 63
Reserved for IEEE 802.1	64 – 255

[a] This is an IETF RFC 2579 DisplayString, with the exception that character codes 0-31 (decimal) are not used.

21.6.5.5 Short MA Name Length

(1 octet) Specifies the length in octets of the Short MA Name field.

Validation Test: The Short MA Name Length in a CCM contains a value greater than or equal to 1.

Validation Test: The Short MA Name Length in a CCM does not indicate that the Short MA Name runs over the 48-octet limit for the MAID.

21.6.5.6 Short MA Name

(Length specified by the Short MA Name Length field) Contains the Short MA Name, in the format specified by the Short MA Name Format field.

21.6.6 Defined by ITU-T Y.1731

(16 octets) The use of this field is defined by ITU-T Y.1731 (2006). A value of 0 should be transmitted in this field.

21.6.7 Optional CCM TLVs

The following TLVs should be included, as allowed by their specifications, in every CCM transmitted:

a) Sender ID TLV (21.5.3);
b) Port Status TLV (21.5.4); and
c) Interface Status TLV (21.5.5).

The following TLV may be included in any CCM transmitted:

d) Organization-Specific TLV (21.5.2).

The Optional CCM TLVs in this subclause may be placed in any order by the MEP transmitting the CCM. The receiving MP shall not depend upon any particular ordering of the Optional CCM TLVs in a CCM for its proper operation.

21.7 Loopback Message and Loopback Reply formats

The format of the LBM and LBR is shown in Table 21-21.

Table 21-21—Loopback Message and Loopback Reply formats

	Octet
Common CFM Header	1 – 4
Loopback Transaction Identifier	5 – 8
Reserved for definition in future versions of the protocol [a]	
Optional LBM/LBR TLVs	First TLV Offset + 5 [b]
End TLV (0)	First TLV Offset + 5, if no Optional LBM/LBR TLVs are present

[a] This field has 0 length in this version 0 of CFM. It is shown in order to stress that additional information can be present in future versions of CFM, and that a version 0 receiver ignores its contents, if present.
[b] Octet 9 for transmitted LBMs.

21.7.1 Flags

(1 octet) In an LBM, the Flags field of the Common CFM Header is set to 0 by the transmitting MP, and is not examined by the receiving MP.

21.7.2 First TLV Offset

(1 octet) The First TLV Offset field of the Common CFM Header in an LBM or LBR is transmitted as 4.

Validation Test: The First TLV Offset field of the Common CFM Header in an LBM or LBR contains a value greater than or equal to 4.

21.7.3 Loopback Transaction Identifier

(4 octets) A MEP copies the contents of the nextLBMtransID variable (20.28.2) to this field.

21.7.4 Optional LBM/LBR TLVs

The following TLV should be included in every LBM transmitted:

a) Sender ID TLV (21.5.3).

The following TLV can be included by management action in any LBM transmitted:

b) Data TLV (21.5.6).

The following TLV may be included in any LBM transmitted:

c) Organization-Specific TLV (21.5.2).

The Optional LBM/LBR TLVs may be placed in any order by the MEP transmitting the LBM.

21.8 Linktrace Message Format

The format of the LTM is shown in Table 21-22.

Table 21-22—Linktrace Message format

	Octet
Common CFM Header	1 – 4
LTM Transaction Identifier	5 – 8
LTM TTL	9
Original MAC Address	10 – 15
Target MAC Address	16 – 21
Reserved for definition in future versions of the protocol [a]	
Additional LTM TLVs	First TLV Offset + 5 [b]
End TLV (0)	First TLV Offset + 5, if no Additional LTM TLVs are present

[a] This field has 0 length in this version 0 of CFM. It is shown in order to stress that additional information can be present in future versions of CFM and that a version 0 receiver ignores its contents, if present.
[b] Octet 22 for transmitted LTMs.

21.8.1 Flags

(1 octet) In the LTM, the Flags field of the Common CFM Header specifies certain options as shown in Table 21-23.

Table 21-23—Linktrace Message Flags field

Mnemonic	Meaning	Bit
UseFDBonly	If set, indicates that only MAC addresses learned in a Bridge's Filtering Database, and not information saved in the MIP CCM Database, is to be used to determine the Egress Port.	8 (MSB)
Reserved	Transmitted as 0, and ignored by the receiver.	7 – 1

21.8.2 First TLV Offset

(1 octet) The First TLV Offset field of the Common CFM Header in an LTM is be transmitted as 17.

Validation Test: The First TLV Offset field of the Common CFM Header in an LTM contains a value greater than or equal to 17.

21.8.3 LTM Transaction Identifier

(4 octets) A MEP copies the contents of the nextLTMtransID variable (20.36.1) to this field.

21.8.4 LTM TTL

(1 octet) The number of hops remaining to this LTM. Decremented by 1 by each Linktrace Responder that handles the LTM. One less than this value is returned in the LTR. If 0 or 1 on input, the LTM is not transmitted to the next hop. If 0 on input, no LTR is returned. The value of the LTM TTL field in the LTM transmitted by the originating MEP is controlled by a managed object [item d) in 12.14.7.4.2]; the default value if none is specified is 64.

21.8.5 Original MAC Address

(6 octets) The MAC address of the MEP that originated the LTM. This can be different from the source MAC address of an LTM because each MIP along the path puts its own MAC address in the source MAC address field, while retaining the Original MAC Address Field.

Validation Test: The Original MAC Address field contains an Individual, and not a Group, MAC address.

21.8.6 Target MAC Address

(6 octets) Specifies an Individual MAC address, the path to which the LTM is intended to trace. Any Individual MAC address can be specified in this field. See also 20.3.1.

Validation Test: The Target MAC Address field contains an Individual, and not a Group, MAC address.

21.8.7 Additional LTM TLVs

The following TLV shall be included in every LTM by each MP originating or forwarding an LTM:

a) LTM Egress Identifier TLV (21.8.8).

The following TLVs may be included in an LTM by each MP originating or forwarding an LTM:

b) Sender ID TLV (21.5.3); and
c) Organization-Specific TLV(s) (21.5.2).

The Additional LTM TLVs may be placed in any order by the MP transmitting the LTM.

21.8.8 LTM Egress Identifier TLV

Type = 7. Identifies the MEP Linktrace Initiator that is originating, or the Linktrace Responder that is forwarding, this LTM. The low-order (highest numbered) six octets contain a 48-bit IEEE MAC address unique to the system in which the MEP Linktrace Initiator or Linktrace Responder resides. The high-order (lowest numbered) two octets contain an integer value sufficient to uniquely identify the transmitted LTM; a constant value (e.g., 0) is sufficient for this purpose, since a Bridge initiates or forwards only a single copy of an LTM. The format of the LTM Egress Identifier TLV is illustrated in Table 21-24.

Table 21-24—LTM Egress Identifier TLV format

	Octet
Type = 7	1
Length	2 – 3
Egress Identifier	4 – 11

NOTE—For most Bridges, the address of any MAC attached to the Bridge will suffice for the low-order six octets, and 0 for the high-order octets. In some situations, e.g., if multiple virtual Bridges utilizing emulated LANs are implemented in a single physical system, the high-order two octets can be used to differentiate among the transmitting entities.

21.9 Linktrace Reply Format

The format of the LTR is shown in Table 21-25.

Table 21-25—Linktrace Reply format

	Octet
Common CFM Header	1 – 4
LTR Transaction Identifier	5 – 8
Reply TTL	9
Relay Action	10
Reserved for definition in future versions of the protocol [a]	
Additional LTR TLVs	First TLV Offset + 5 [b]
End TLV (0)	First TLV Offset + 5, if no Additional LTR TLVs are present

[a] This field has 0 length in this version 0 of CFM. It is shown in order to stress that additional information can be present in future versions of CFM, and that a version 0 receiver ignores its contents, if present.
[b] Octet 11 for transmitted LTRs.

Validation Test: The Reply Ingress TLV (21.9.8), the Reply Egress TLV (21.9.9), or both, are present.

21.9.1 Flags

(1 octet) In the LTR, the Flags field of the Common CFM Header specifies certain options as shown in Table 21-26.

Table 21-26—Linktrace Reply Flags field

Mnemonic	Meaning	Bit
UseFDBonly	Copied from LTM.	8 (MSB)
FwdYes	The LTM was (1) or was not (0) forwarded.	7
TerminalMEP	The MP reported in the Reply Egress TLV (Reply Ingress TLV, if the Reply Egress TLV is not present) is a MEP.	6
Reserved	Copied from LTM.	5 – 1

21.9.2 First TLV Offset

(1 octet) The First TLV Offset field of the Common CFM Header in an LTR is transmitted as 6.

Validation Test: The First TLV Offset field of the Common CFM Header in an LTR contains a value greater than or equal to 6.

21.9.3 LTR Transaction Identifier

(4 octets) The value from the LTM Transaction Identifier field in the LTM that triggered the transmission of this LTR.

21.9.4 Reply TTL

(1 octet) One less than the value from the LTM TTL field in the LTM that triggered the transmission of this LTR. If the LTM TTL field contained a 0, no LTR is transmitted.

21.9.5 Relay Action

(1 octet) Reports how the data frame targeted by the LTM would be passed through the MAC Relay Entity to the Egress Bridge Port, as shown in Table 21-27.

Table 21-27—Relay Action field values

Mnemonic	Relay action	Value
RlyHit	The LTM reached an MP whose MAC address matches the target MAC address.	1
RlyFDB	The Egress Port was determined by consulting the Filtering Database [item a) in 20.42.1.2].	2
RlyMPDB	The Egress Port was determined by consulting the MIP CCM Database [item b) in 20.42.1.2].	3

Validation Test: The value in the Relay Action field is one of the values enumerated in 21.9.5.

21.9.6 Additional LTR TLVs

The following TLV shall be included in an LTR according to 20.3.2:

a) LTR Egress Identifier TLV (21.9.7).

One or both of the following TLVs shall be included in an LTR according to 20.3.2:

b) Reply Ingress TLV (21.9.8); or
c) Reply Egress TLV (21.9.9).

The following TLVs may be included in an LTR:

d) Sender ID TLV (21.5.3);
e) Organization-Specific TLV (21.5.2).

The Additional LTR TLVs may be placed in any order by the MP transmitting the LTR. The receiving MEP shall not depend upon any particular ordering of the Additional LTR TLVs in an LTR for its proper operation.

21.9.7 LTR Egress Identifier TLV

Type = 8. Identifies the source and destination of the LTM that triggered transmission of this LTR. The format of the LTR Egress Identifier TLV is illustrated in Table 21-28.

Table 21-28—LTR Egress Identifier TLV format

	Octet
Type = 8	1
Length	2 – 3
Last Egress Identifier	4 – 11
Next Egress Identifier	12 – 19

21.9.7.1 Last Egress Identifier

(8 octets) Identifies the MEP Linktrace Initiator that originated, or the Linktrace Responder that forwarded, the LTM to which this LTR is the response. This field takes the same value as the LTM Egress Identifier TLV of that LTM, or the value 0, if no LTM Egress Identifier TLV was present in that LTM.

21.9.7.2 Next Egress Identifier

(8 octets) Identifies the Linktrace Responder that transmitted this LTR, and can forward the LTM to the next hop. This field takes the same value as the LTM Egress Identifier TLV of the forwarded LTM, if any. If the FwdYes bit of the Flags field is false, the contents of this field are undefined, i.e., any value can be transmitted, and the field is ignored by the receiver.

21.9.8 Reply Ingress TLV

Type = 5. The format of the Reply Ingress TLV is shown in Table 21-29. The Reply Ingress TLV is transmitted if and only if the Bridge Port on which the LTM was received has an MP in the MA specified by the LTM.

Table 21-29—Reply Ingress TLV format

	Octet
Type = 5	1
Length	2 – 3
Ingress Action	4
Ingress MAC Address	5 – 10
Ingress Port ID Length	11
Ingress Port ID Subtype	12
Ingress Port ID	13 – (Length + 3)

Validation Test: The value in the Ingress Action field is one of the values enumerated in 21.9.8.1.

Validation Test: The Ingress MAC Address field contains an Individual, and not a Group, MAC address.

Validation Test: The Length field is either 7, indicating that no Ingress Port ID field is present, or is (9 + the Ingress Port ID Length field) or larger, and thus contains the Ingress Port ID field.

21.9.8.1 Ingress Action

(1 octet) Reports how the data frame targeted by the LTM would be received on the receiving MP, as shown in Table 21-30.

Table 21-30—Ingress Action field values

Mnemonic	Ingress action	Value
IngOK	The target data frame would be passed through to the MAC Relay Entity.	1
IngDown	The Bridge Port's MAC_Operational parameter is false. [a]	2
IngBlocked	The target data frame would not be forwarded if received on this Port due to active topology enforcement (8.6.1). (See Figure 22-1 to see how this is possible.)	3
IngVID	The ingress port is not in the member set of the LTM's VID, and ingress filtering is enabled, so the target data frame would be filtered by ingress filtering (8.6.2).	4

[a] This value could be returned, for example, by an operational Down MEP that has another Down MEP at a higher MD Level on the same Bridge Port that is causing the Bridge Port's MAC_Operational parameter to be false.

21.9.8.2 Ingress MAC Address

(6 octets) The MAC address that can be used as the Destination MAC address of an LBM intended for the ingress MP.

21.9.8.3 Ingress Port ID Length

(1 octet) The length, in octets, of the Ingress Port ID field. Cannot be 0. If the Length field of the Reply Ingress TLV is less than 8, then the Ingress Port ID Length field, Ingress Port ID Subtype field, and Ingress Port ID field are not present.

21.9.8.4 Ingress Port ID Subtype

(1 octet) Identifies the format of the Ingress Port ID field. Format specified by IEEE Std 802.1AB-2005, 9.5.3.2. If the Length field of the Reply Ingress TLV is less than 8, then the Ingress Port ID Subtype field is not present.

21.9.8.5 Ingress Port ID

(Length specified by the Ingress Port ID Length field) Identifies the Ingress Port within the chassis identified in the Sender ID TLV, in the format specified by IEEE Std 802.1AB-2005, 9.5.3.3. If the Length field of the Reply Ingress TLV is less than 8, then the Ingress Port ID field is not present.

21.9.9 Reply Egress TLV

Type = 6. The format of the Reply Egress TLV is shown in Table 21-31. The Reply Egress TLV is included in the LTR if and only if the Bridge Port on which the LTM would be relayed could be identified, and is an MP in the MA specified by the LTM.

Validation Test: The value in the Egress Action field is one of the values enumerated in 21.9.9.1.

Table 21-31—Reply Egress TLV format

	Octet
Type = 6	1
Length	2 – 3
Egress Action	4
Egress MAC Address	5 – 10
Egress Port ID Length	11
Egress Port ID Subtype	12
Egress Port ID	13 – (Length + 3)

Validation Test: The Egress MAC Address field contains an Individual, and not a Group, MAC address.

Validation Test: The Length field is either 7, indicating that no Egress Port ID field is present, or is (9 + the Egress Port ID Length field) or larger, and thus contains the Egress Port ID field.

21.9.9.1 Egress Action

(1 octet) Reports how the data frame targeted by the LTM would be passed through Egress Bridge Port, as shown in Table 21-32.

Table 21-32—Egress Action field values

mnemonic	Egress action	value
EgrOK	The targeted data frame would be forwarded.	1
EgrDown	The Egress Port can be identified, but that Bridge Port’s MAC_Operational parameter is false.	2
EgrBlocked	The Egress Port can be identified, but the data frame would not pass through the Egress Port due to active topology management (8.6.1), i.e., the Bridge Port is not in the Forwarding state.	3
EgrVID	The Egress Port can be identified, but the Bridge Port is not in the LTM’s VID’s member set, so would be filtered by egress filtering (8.6.4).	4

21.9.9.2 Egress MAC Address

(6 octets) The MAC address that can be used as the Destination MAC address of an LBM intended for the egress MP.

21.9.9.3 Egress Port ID Length

(1 octet) The length, in octets, of the Egress Port ID field. Cannot be 0. If the Length field of the Reply Egress TLV is less than 8, then the Egress Port ID Length field, Egress Port ID Subtype field, and Egress Port ID field are not present.

21.9.9.4 Egress Port ID Subtype

(1 octet) Identifies the format of the Egress Port ID field. Format specified by IEEE Std 802.1AB-2005, 9.5.3.2. If the Length field of the Reply Egress TLV is less than 8, then the Egress Port ID Subtype field is not present.

21.9.9.5 Egress Port ID

(Length specified by the Egress Port ID Length field) Identifies the Egress Port within the chassis identified in the Sender ID TLV. Format specified by IEEE Std 802.1AB-2005, 9.5.3.3. If the Length field of the Reply Egress TLV is less than 8, then the Egress Port ID field is not present.

Insert a new Clause 22 as follows.

22. Connectivity Fault Management in systems

The relationships among Maintenance Points (MPs), and between the MPs and the other entities in a Bridge, are configurable. This configuration determines how CFM is used in a network and determines what parts of the Bridges and LANs comprising a network are protected by a Maintenance Association (MA). This clause provides both specifications and explanatory text to assist the implementor and system administrator to make efficient use of CFM. This clause:

a) Specifies how the EISS Multiplex Entity (6.15) is used to support MPs for multiple MAs associated with service instances supported by EISS-SAPs (22.1);
b) Specifies the creation of MPs in a Bridge (22.2);
c) Suggests methods for the efficient assignment of MD Levels to Maintenance Domains (22.3);
d) Specifies the use of MPs in stations (22.4);
e) Clarifies issues related to the scaling of resources required to run CFM (22.5);
f) Shows how MPs can be placed in Provider Bridges (22.6);
g) Suggests methods for using CFM in the enterprise environment, as opposed to the provider environment (22.7); and
h) Provides guidance for the implementation of CFM in Bridges designed before the development of this standard (22.8).

NOTE—Clause 18 introduces the principles of CFM operation and the network architectural concepts that support it. Clause 19 breaks down the CFM protocol entities into their components. Clause 20 specifies the protocols operated by the components of each MP, and Clause 21 specifies the PDU formats used by those protocols.

22.1 CFM shims in Bridges

CFM is instantiated by configuring a Maintenance Association (MA), which maintains a single service instance. In the case that multiple MAs are configured on a single Bridge Port, the vlan_identifier parameter of the EISS and the MD Level in the CFM PDUs are the means by which CFM PDUs for the various MAs are multiplexed. Non-CFM data frames are distinguished only by vlan_identifier; the particular service instance to which a non-CFM data frame belongs, if multiple service instances share the same VID, distinguished only by MD Level, is undetermined.

22.1.1 Preliminary positioning of Maintenance Points

Figure 22-1 illustrates the method by which MPs are associated with particular service instances. The EISS Multiplex Entity (6.15) distributes frames across an array of ISS or EISS-SAPs, each of which can have a "column" of MPs at different MD Levels and orientations (Up or Down). Several MEPs and MHFs are shown, distinguished by VID and MD Level in this manner.

An MP that has its Passive SAP closer to the MAC Relay Entity than its Active SAP is called a "Down MEP" or "Down MHF," because it points down, away from the MAC Relay Entity, in Figure 22-1. An MP that has its Passive SAP further away from the MAC Relay Entity than its Active SAP is called an "Up MEP" or "Up MHF." The Bridge Port in Figure 22-1 is configured with five Up MEPs, three Up MHFs, three Down MHFs, and three Down MEPs. A single Linktrace Output Multiplexer (LOM) is positioned below the per-VID Down MEPs.

Figure 22-1 illustrates a Down MEP, at MD Level 1, configured on the ISS below the Bridge Port connectivity entity. The service instance its MA protects encompasses all data frames passing through the Bridge Port, no matter what VLAN they are associated with.

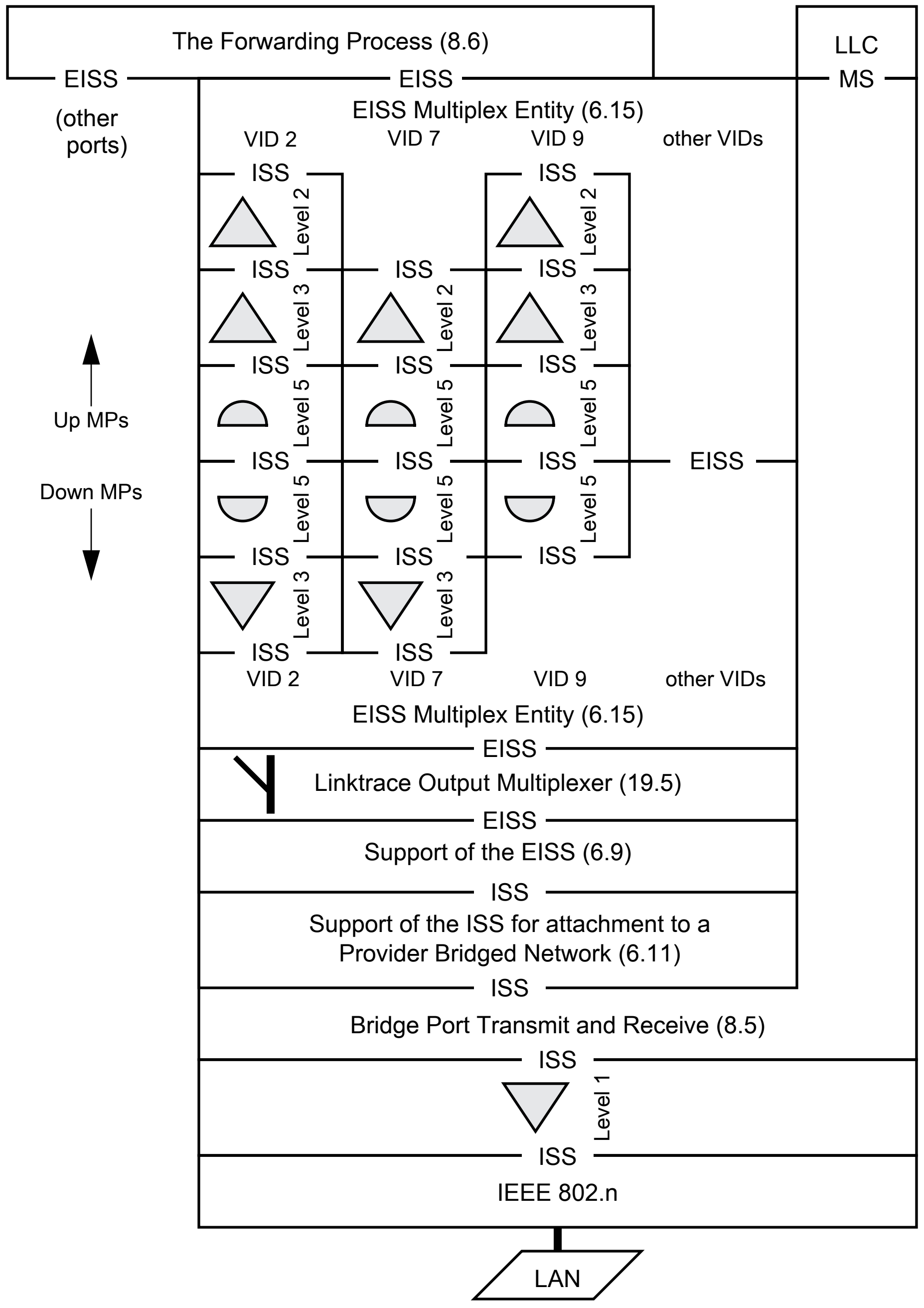
The Forwarding Process (8.6)
LLC
MS
EISS
EISS
(other ports)
EISS Multiplex Entity (6.15)
VID 2
VID 7
VID 9
other VIDs
ISS
Level 2
Level 3
Level 5
Level 5
Level 3
Up MPs
Down MPs
EISS
VID 2
VID 7
VID 9
other VIDs
EISS Multiplex Entity (6.15)
EISS
Linktrace Output Multiplexer (19.5)
EISS
Support of the EISS (6.9)
ISS
Support of the ISS for attachment to a Provider Bridged Network (6.11)
ISS
Bridge Port Transmit and Receive (8.5)
ISS
Level 1
ISS
IEEE 802.n
LAN

Figure 22-1—MEPs and MIPs distinguished by VID (incomplete picture)

22.1.2 CFM and the Forwarding Process

Figure 22-1 is incomplete, because it does not illustrate the relationship of CFM to the various components of the Forwarding Process (8.6). These components are illustrated in Figure 8-9 of IEEE Std 802.1Q-2005. We can notice two facts about Figure 8-9:

a) With the exception of Frame filtering (8.6.3), every one of the functions in Figure 8-9 is making decisions about the flow of frames on a single Bridge Port. Frame filtering is concerned with all of the Bridge’s Ports.

b) The parameters passed from component to component in Figure 8-9 are exactly those of the EISS, and therefore each of the components in Figure 8-9 could be described as a shim.

From these facts, we can reconstruct Figure 8-9 in a manner that is functionally identical, but is better suited to illustrate the relationship between CFM and the Forwarding Process. This reconstruction is shown in Figure 22-2. In this figure, the per-port functions are separated into the Port filtering entities (8.6.1, 8.6.2, and 8.6.4) and the Queuing entities (8.6.5, 8.6.6, 8.6.7, and 8.6.8).

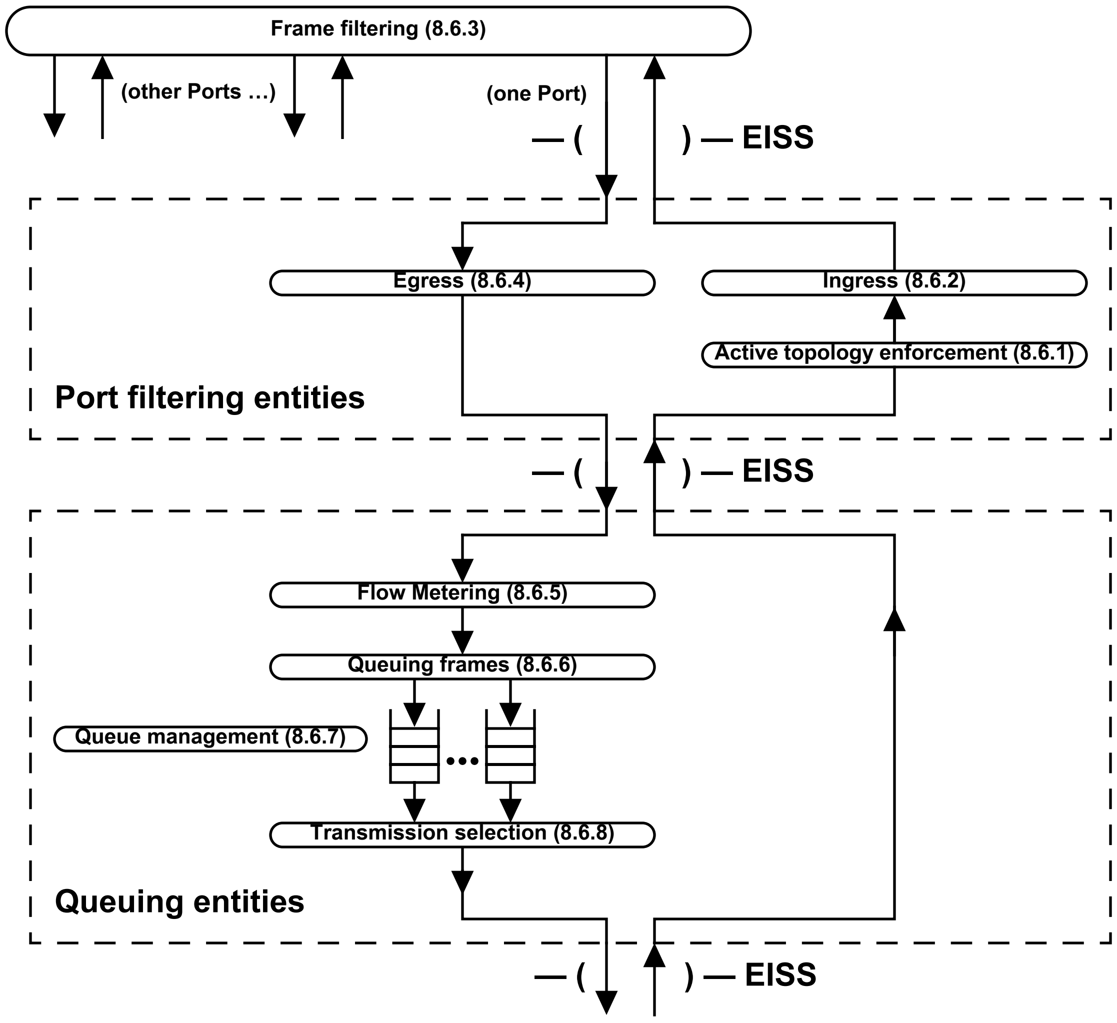

Figure 22-2—Alternate view of Forwarding process

22.1.3 Up/Down separation of Maintenance Points

An additional transformation to Figure 22-1 is required before the relationship between the Forwarding Process and CFM can be shown clearly. By adding a further pair of EISS Multiplex Entities between the Up MPs and the Down MPs in Figure 22-1, the entire set of Up MPs can be shown as a single shim, as can the entire set of Down MPs. This transformation is illustrated in Figure 22-3.

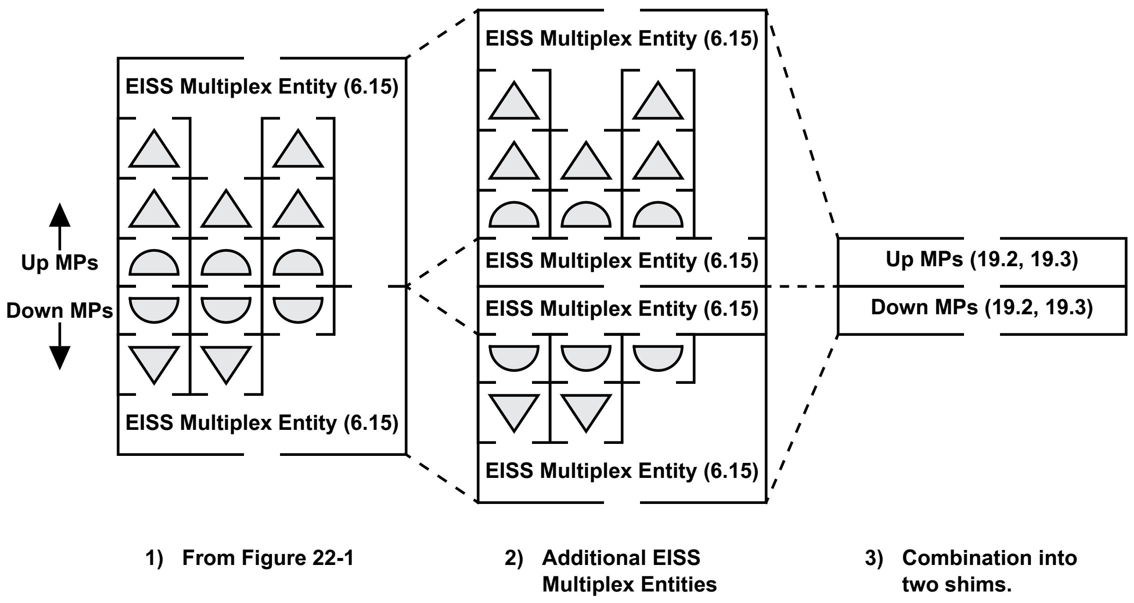
EISS Multiplex Entity (6.15)
Up MPs
Down MPs
EISS Multiplex Entity (6.15)
EISS Multiplex Entity (6.15)
EISS Multiplex Entity (6.15)
EISS Multiplex Entity (6.15)
EISS Multiplex Entity (6.15)
Up MPs (19.2, 19.3)
Down MPs (19.2, 19.3)
1) From Figure 22-1
2) Additional EISS Multiplex Entities
3) Combination into two shims.

Figure 22-3—Combining per-VLAN Maintenance Points into two shims

Figure 22-4 combines Figure 22-1 with Figure 22-2, and shows the relationship of CFM to the various components of the Forwarding Process. CFM PDUs are treated by the Active topology enforcement entity (8.6.1) in the same manner as ordinary data. MEPs and MHFs source and sink CFM PDUs. The flow of data frames through a Bridge Port is blocked by the Port filtering entities, Active topology enforcement (8.6.1), Ingress (8.6.2), and Egress (8.6.4). Thus, the five Up MEPs of Figure 22-4 can transmit and receive CFM PDUs through the Bridge even when the Bridge Port is blocked. Similarly, the two Down MEPs in Figure 22-4 operate normally, transmitting CFM PDUs to and receiving them from the LAN, even if the Bridge Port is blocked. However, no frames, neither CFM PDUs nor ordinary data, can pass through the Port filtering entities when the VLAN to which those frames or CFM PDUs belong is filtered.

Also shown in Figure 22-4 are the positions of the Domain SAP (DoSAPs) and ISAPs introduced in Clause 18. The DoSAP provides access to a service instance (an instance of the MAC Service) and exists independently of any MA that might be created to protect it. The DoSAPs and ISAPs, therefore, can be associated with the specific inter-shim SAPs at the points indicated in Figure 22-4. The DoSAPs and ISAPs are attached to the Port filtering entities, because that is where the decisions are made on whether frames belonging to the service instance are or are not allowed to pass through the Bridge Port.[13] It does not matter to which side of the EISS Multiplex Entity one considers the DoSAPs and ISAPs to be attached. Except for the topmost Down MEP in each stack of Down MPs, the DoSAP is not coincident with the Passive SAP of the MEP protecting the DoSAP's service instance; the DoSAP is separated from its MA by the MHFs and/or MEPs at higher MD Levels. Even for that topmost Down MEP, an MHF for a higher MD Level can separate that MEP from its DoSAP. This gap is illustrated in Figure 22-5 and Figure 22-6, where the service instance, DoSAP to DoSAP, is shown in gray, but the MEPs protecting that service are separated from the DoSAPs by MHFs belonging to a service instance at a higher MD Level.

[13]The DoSAP is not attached to an MP's Passive SAP because the DoSAP and its associated service instance exist whether or not the service instance is protected by an MA.

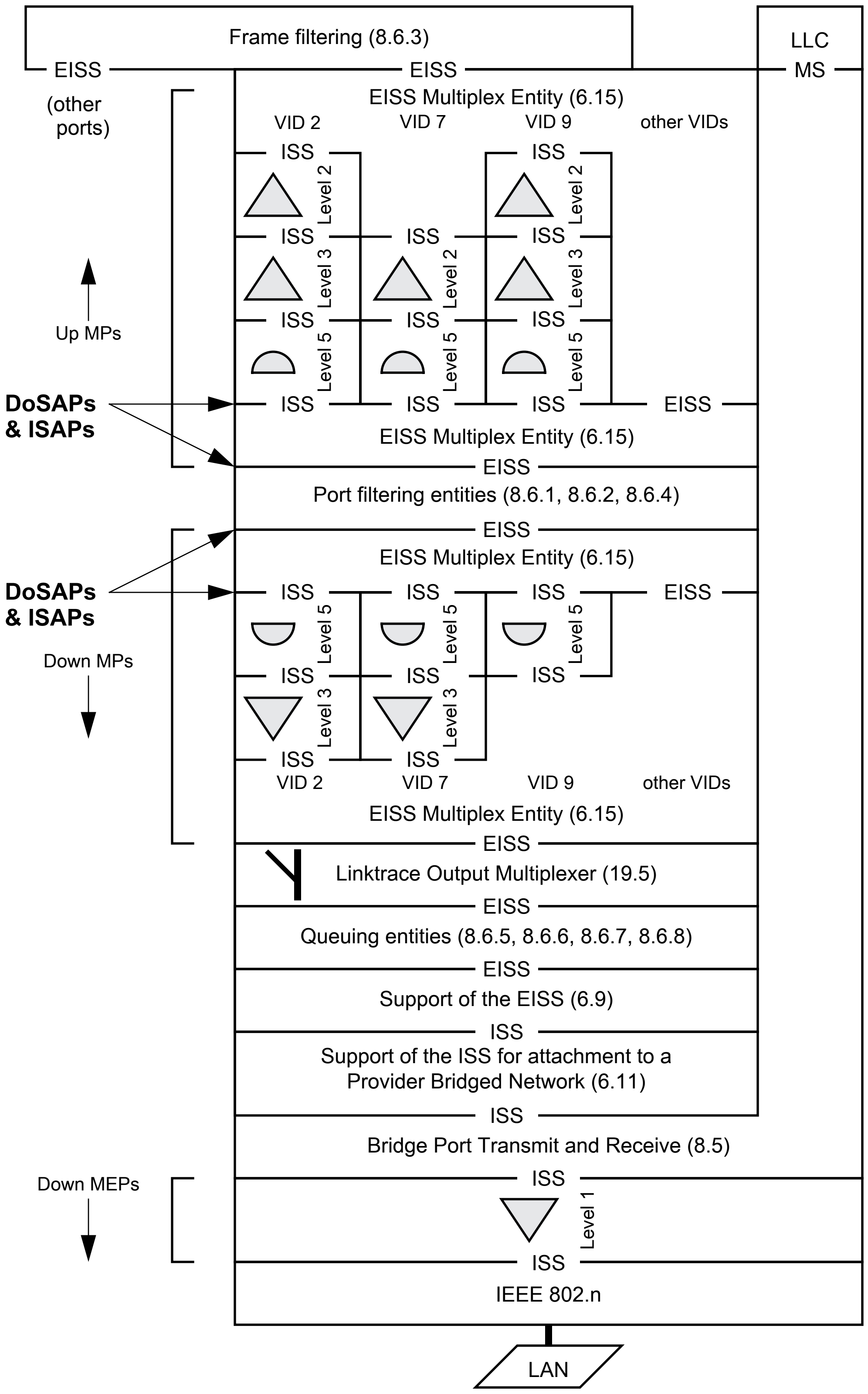
Frame filtering (8.6.3)
LLC
EISS
EISS
MS
(other ports)
EISS Multiplex Entity (6.15)
VID 2
VID 7
VID 9
other VIDs
ISS
Level 2
Level 3
Level 5
Up MPs
DoSAPs & ISAPs
EISS
EISS Multiplex Entity (6.15)
Port filtering entities (8.6.1, 8.6.2, 8.6.4)
Down MPs
Linktrace Output Multiplexer (19.5)
Queuing entities (8.6.5, 8.6.6, 8.6.7, 8.6.8)
Support of the EISS (6.9)
Support of the ISS for attachment to a Provider Bridged Network (6.11)
Bridge Port Transmit and Receive (8.5)
Down MEPs
Level 1
IEEE 802.n
LAN

Figure 22-4—More complete picture of Maintenance Point placement in a Bridge Port

In the “stack” of MPs on a single Bridge Port, the following rules apply:

a) All Up MEPs belonging to MAs that are attached to specific VIDs are placed between the Frame filtering entity (8.6.3) and the Port filtering entities (8.6.1, 8.6.2, and 8.6.4). Separately for each VLAN, there can be from zero to eight Up MEPs, ordered by increasing MD Level, from Frame filtering towards Port filtering.
b) For each MA there can be at most one MIP, consisting of an Up MHF just above, and a Down MHF just below, the Port filtering entities. Both MHFs are at the same MD Level, higher than the highest MEP on the same VLAN, if any;
c) Up to eight Down MEPs belonging to MAs that are attached to the same list of VIDs are placed between the Down MHF, if any, and the Queuing entities (8.6.5, 8.6.6, 8.6.7, and 8.6.8). They are placed in order of decreasing MD Level, from the Port filtering entities to the Queuing entities, including any Down MHF.
d) An LOM is positioned below the Down MPs belonging to MAs that are attached to specific VIDs to handle forwarded LTMs.
e) Up to eight Down MEPs, ordered from highest MD Level to lowest, belonging to MAs not attached to any specific VID, can be placed below Bridge Port connectivity (8.5.1). They are ordered by decreasing MD Level going away from the MAC Relay Entity (See also 22.1.8).
f) All of the Down MEPs below Bridge Port connectivity have lower MD Levels than any Down MP attached to a VID that is emitted untagged by the Support of the EISS.
g) Any shim that introduces an EtherType tag insulates the MPs above and below it, removing any MD Level restriction between the separated groups of MPs.

A single Bridge Port could, in theory, have MHFs for multiple service instances at different MD Levels. To do so, however, would require a vector of 4094 MD Level registers instead of one register to identify which CFM PDUs need to be deflected to the Bridge’s Higher Layer Entities. This capability is optional.

22.1.4 Service instances over multiple Bridges

Figure 22-4 completes the sequence of expansions starting with Figure 18-5 and continuing with Figure 18-6 and Figure 18-7. In this sequence, Figure 22-4 illustrates Bridge Port b of Bridge 2 in Figure 18-5 through Figure 18-7. The service instances illustrated in those three figures all use VID 9 in Figure 22-4. The Up MEPs for VID 9 at MD Levels 3 and 2 are the Up MEPs protecting the provider MD Level and the operator MD Level service instances, respectively, and the pair of MHFs at MD Level 5 on VID 9 provide a customer MD Level MIP. Figure 22-4 also illustrates two other sets of service instances sharing the same Bridge Port using VIDs 2 and 7.

Additional views of service instances spanning two Bridges are shown in Figure 22-5 and Figure 22-6. Figure 22-5 shows a service instance protected by Up MPs, and Figure 22-6 shows a service instance protected by Down MPs.

22.1.5 Multiple VID service instances

A service instance can be implemented using multiple VIDs. It is always the case that the VLAN tag uniquely determines the service instance to which a given frame belongs. An MA with multiple VIDs does not require any more MIPs or MEPs than an MA with a single VID. All of the VIDs belonging to a given service instance share the same MA. That is, one MAID is used for all CFM PDUs on the various VIDs in the same service instance and Maintenance Domain. The choice of which VID to use to tag a given CFM PDU depends on the OpCode, as specified in Clause 20.

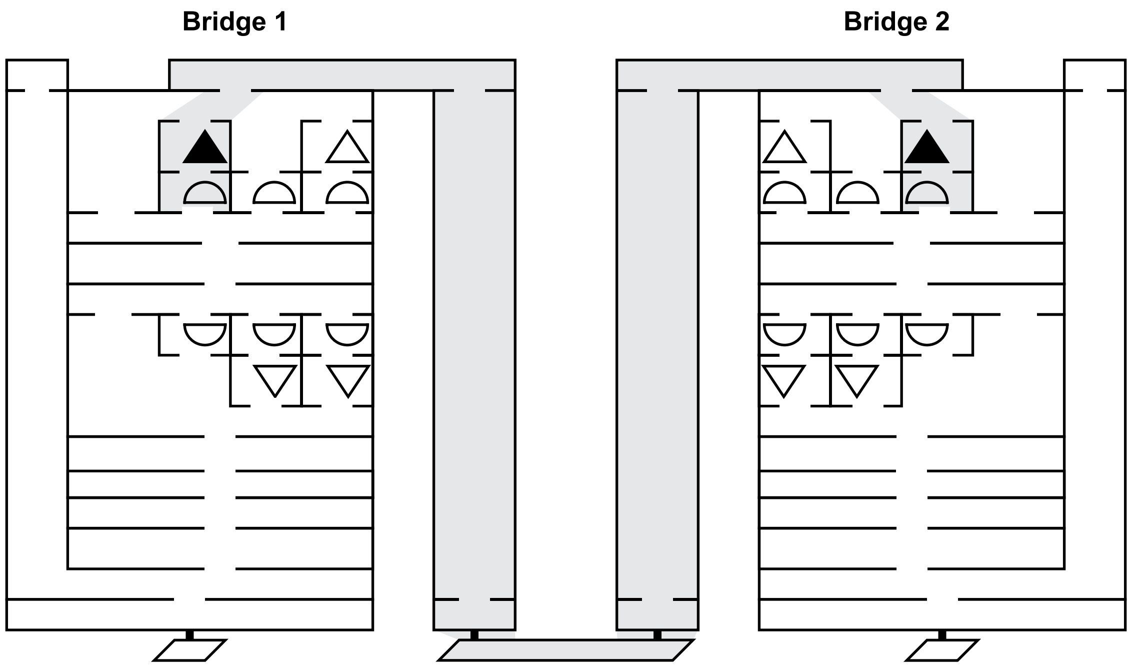

Figure 22-5—Service instance spanning two Bridges protected by Up MPs

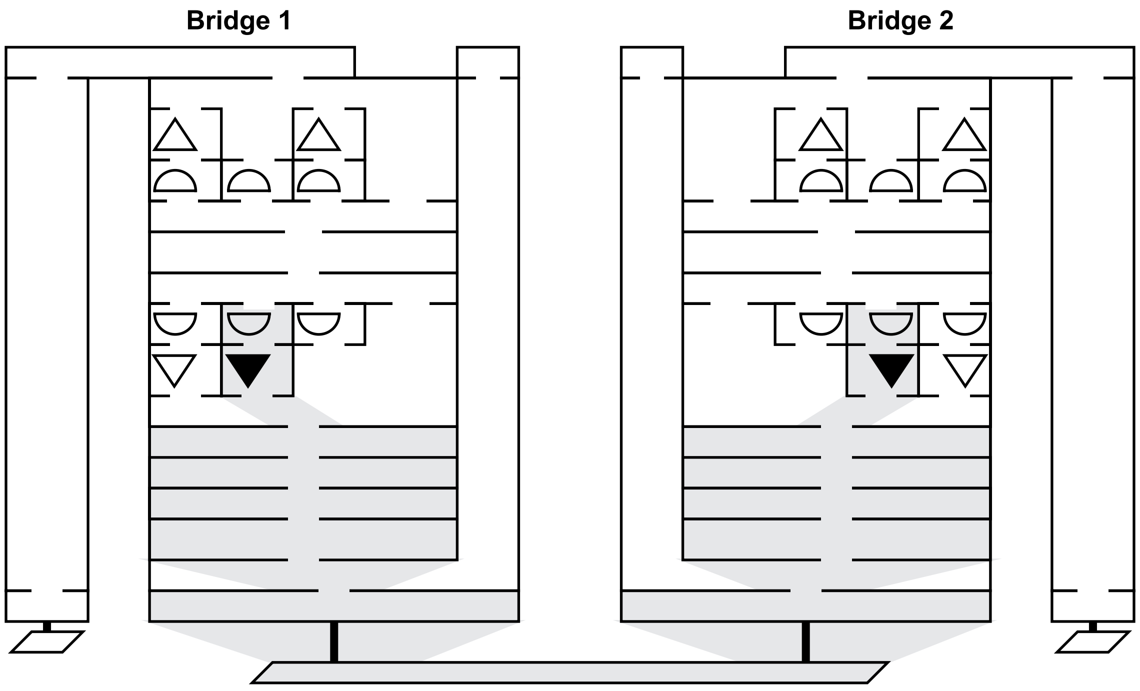

Figure 22-6—Service instance spanning two Bridges protected by Down MPs

22.1.6 Untagged CFM PDUs

A MEP placed anywhere below the Support of the EISS entity (6.9) or the Bridge Port connectivity entity (8.5.1) differs in three important respects from a MEP placed above the Support of the EISS entity:

a) The lower position would not have the Queuing entities available to it. For example, the reflection of a high-speed stream of LBMs (as LBRs) could seriously disrupt the flow of data frames, and thus bring into question the utility of the Queuing entities, CFM, or both.
b) A MEP in the lower position is necessarily untagged; the tagging functions are performed in the Support of the EISS entity.
c) A MEP in the lower position protects all data passing through the Bridge Port; it does not separately protect the different VLANs.

Item c) does not mean that the lower position is the only one that can protect all of the traffic on a single link. MEPs placed above the Queuing entities can be associated with a VLAN whose VID is the PVID, for which this Bridge Port belongs to the untagged set. That VLAN can be statically configured in the member set to not be allowed to pass through the Frame filtering entity (8.6.3). MEPs so configured can then transmit and receive only untagged frames, and thus protect all of the data on the link. Their untagged CFM PDUs can be visible to complex “links” such as Provider Bridged Networks, even though the C-VLAN tagged CFM PDUs are not visible inside the Provider Bridged Network.

The option to create MEPs below the Bridge Port connectivity entity cannot be omitted, however, because of considerations given in 22.1.8. This subclause also explains why only MEPs, and not MHFs, can be placed below the Bridge Port connectivity entity.

22.1.7 Maintenance Points and non-VLAN aware Bridges

The positioning of Maintenance Points Bridges that are not VLAN aware is illustrated in Figure 22-7. Because there are no VLANs, no EISS Multiplex Entity is needed. However, the Up MPs are still between the Frame filtering function and the Port filtering entities, so the transformation introduced in 22.1.2 is still retained.

Down MPs can be placed either above or below the Queuing entities and Bridge Port connectivity in Figure 22-7; the difference between the two locations is that:

a) In the upper location, frame queues are available to CFM, so that CFM frames and data frames can be scheduled appropriately for output; and
b) Placing a Down MEP in the lower location protects the LLC path used by the Higher Layer Entities, e.g., BPDUs, as well as the normal data forwarding path.

22.1.8 Maintenance Points and other standards

The relationship of Maintenance Points and the shims defined by two other standards, IEEE Std 802.3, Clause 43, Link Aggregation, and IEEE Std 802.1AE, MAC Security, is illustrated in Figure 22-8. This figure shows Down MEPs positioned both above and below Link Aggregation, and illustrates a number of points.

MPs could, in theory, be placed either above or below the MAC Security shim. No loss of data integrity beyond the protected link would be incurred by placing CFM below MAC Security; the only attacks that could be generated would be variants of a denial of service attack. However, there are several reasons for restricting MPs to positions above the MAC Security shim:

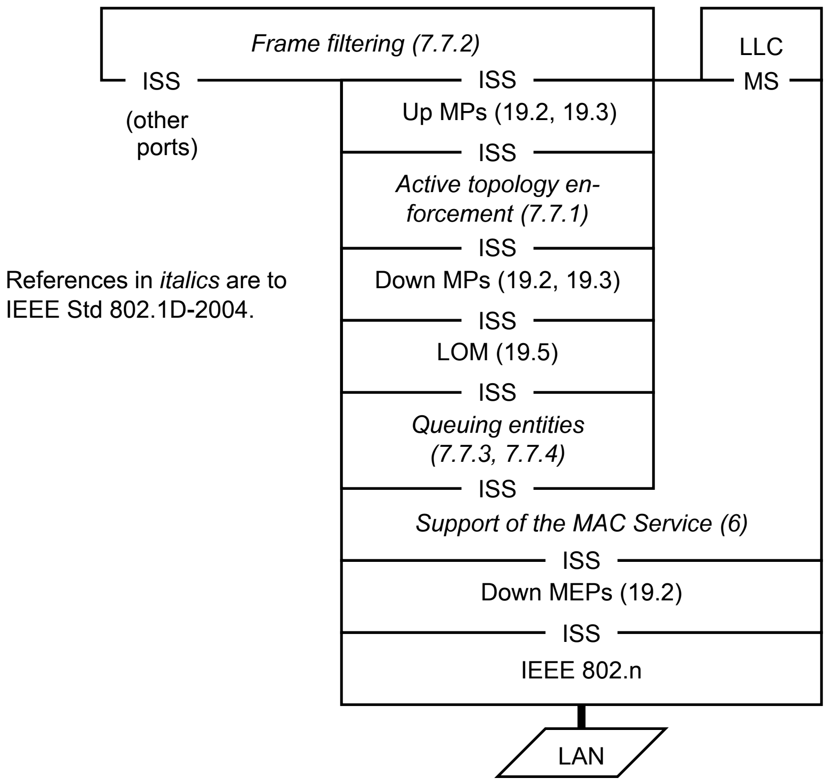

Figure 22-7—Maintenance Point placement in a non-VLAN aware Bridge Port

a) The denial of service attacks available by the introduction of bogus CFM PDUs have more potential for disrupting management operations, in more complex ways, than do attacks based on other unprotected protocols, namely those defined by IEEE Std 802.3. For example, bogus cross connect CCMs could be introduced, leading to undesirable interactions between different providers' management systems.
b) MAC Security ensures that all data frames are legitimate, in the sense that they were last transmitted by a trusted device. Allowing the introduction of unprotected CFM PDUs violates the principle that data frames and CFM PDUs follow the same paths, distinguished only by the CFM protocol encapsulation.
c) Allowing the CFM and MAC Security shims to be interchanged by management operations could significantly complicate the implementation of a Bridge, with no commensurate benefit to the user.

As described in 22.1.6, placing an MP below the Queuing entities, whether above or below the Link Aggregation shim, is not desirable, as that MP is not able to use the Queuing entities. For this reason, it is preferable to use a Down MEP that is attached to a specific VLAN, and is thus above the Queuing entities, to protect the link.

When Link Aggregation is employed, Down MEPs attached to no VLAN can be useful for protecting the individual IEEE 802.3 LANs. The method used in Link Aggregation to select the outgoing IEEE 802.3 link is undefined. It is not necessarily possible for an MP above the Queuing entities to protect all of a Bridge Port's data; CCM PDUs could take one IEEE 802.3 link, and data frames another. Therefore, provision is made in this standard for configuring Down MEPs in an MA attached to no VLAN below the Bridge Port connectivity entity, and below Link Aggregation, when employed. The following considerations are important when configuring Down MEPs:

d) It is best to disable the Bridge Port before issuing a high-speed stream of LBRs (relative to the physical link speed), either from or towards a Down MEP attached to no VLAN; otherwise, the Queuing entities might be unable to achieve any guarantees for throughput or latency.
e) The rate of CCMs generated by Down MEPs attached to VLANs should be taken into account when configuring the Queuing entities to achieve particular guarantees for throughput or latency.
f) If Link Aggregation is present, Down MEPs attached to no VLAN can only be configured below Link Aggregation. Allowing configuration both above and/or below Link Aggregation could significantly complicate an implementation, while providing little added utility to the user.
g) The creation of MHFs below Link Aggregation presents two problems that preclude it from this standard:
    1) IEEE Std 802.3 provides no means for Link Aggregation to use a destination MAC address to direct a CFM PDU down the correct link to reach a particular MP at one or the other end of that link.
    2) The utility of isolating the short segment between the Port filtering entities and the links being aggregated, which is completely internal to a Bridge, is very limited, especially considering the wide range of methods employed for implementing Link Aggregation.

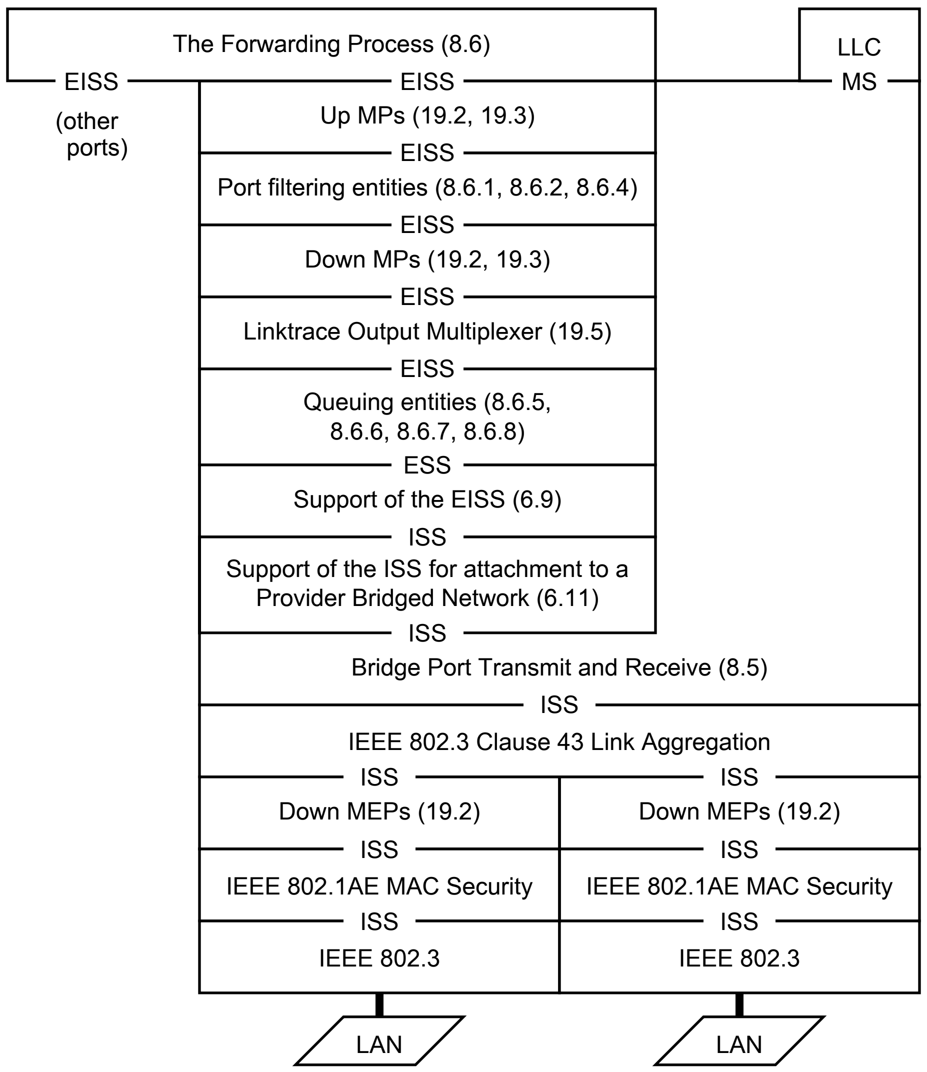
The Forwarding Process (8.6)
LLC
EISS
EISS
MS
(other ports)
Up MPs (19.2, 19.3)
EISS
Port filtering entities (8.6.1, 8.6.2, 8.6.4)
EISS
Down MPs (19.2, 19.3)
EISS
Linktrace Output Multiplexer (19.5)
EISS
Queuing entities (8.6.5, 8.6.6, 8.6.7, 8.6.8)
ESS
Support of the EISS (6.9)
ISS
Support of the ISS for attachment to a Provider Bridged Network (6.11)
ISS
Bridge Port Transmit and Receive (8.5)
ISS
IEEE 802.3 Clause 43 Link Aggregation
ISS
ISS
Down MEPs (19.2)
Down MEPs (19.2)
ISS
ISS
IEEE 802.1AE MAC Security
IEEE 802.1AE MAC Security
ISS
ISS
IEEE 802.3
IEEE 802.3
LAN
LAN

Figure 22-8—Maintenance Point placement relative to other standards

22.1.9 CFM and IEEE 802.3 Clause 57 OAM

The following differences between CFM and IEEE 802.3 Clause 57 OAM are relevant to making a decision whether to employ one, the other, or both of these protocols.

a) IEEE 802.3 OAM has provision for a loopback mode in which all frames are returned to the sender. CFM does not.

NOTE—IEEE 802.3 OAM confines loopbacked frames to the individual IEEE 802.3 LAN, effectively disabling any attached Bridge Port. Use of simple frame loopback through a Bridged Local Area Network could be expected to seriously impact the operation of the network.

b) IEEE 802.3 OAM has provision for acquiring statistics from a remote interface. CFM does not.
c) IEEE 802.3 OAM has provision for generating error conditions when configured thresholds are exceeded for parameters such as the number of frames received with CRC errors. CFM does not.
d) IEEE 802.3 OAM has provision for organizations other than IEEE 802 to define both new TLVs and new operation codes using OUIs. CFM allows only TLVs to be defined using OUIs.
e) IEEE 802.3 OAM performs a continuity check once per second. CFM can be programmed to perform the continuity check over a wide range of intervals.
f) IEEE 802.3 OAM is defined only for IEEE 802.3 media. CFM can be used over any medium, or Bridged Network, that can carry IEEE 802 frames.
g) IEEE 802.3 OAM protects a single link. CFM can protect a single link or a network of Bridged Networks, over multiple nested ranges.
h) CFM can protect the connectivity of shared media. IEEE 802.3 OAM has no such capability.
i) CFM can perform in-band loopback tests at any rate up to the physical line rate, during normal operations. IEEE 802.3 OAM has no such capability.
j) CFM can perform Linktrace functions through a Bridged LAN. IEEE 802.3 OAM has no such capability.
k) CFM can detect unintentional connectivity, as well as lack of connectivity. IEEE 802.3 OAM cannot.

22.2 Maintenance Entity creation

MPs are created using the Managed Objects described in 12.14. Before a MEP or MHF can be created, either an entry in the Default MD Level managed object, or the Maintenance Domain and MA to which the MEP or MHF belongs, are created. In this context, a Default MD Level managed object, Maintenance Domain, or an MA are strictly managed objects in a single Bridge. In order to realize the network-wide definitions of Maintenance Domains and MAs given in 18.1 and 18.2, the Bridges in a Maintenance Domain are configured with identical information in these managed objects. During transitional periods when this information is changing, Fault Alarms will likely be generated.

NOTE—The correct operation of CFM does not depend on the configuration restrictions in this subclause. Rather, violating these restrictions is likely to generate Fault Alarms that might or might not be erroneous. In other words, if no Fault Alarms are generated, then all monitored MAs have connectivity. If any Fault Alarm is generated, then either connectivity has been lost or the network is improperly configured. Determining which of these two is the cause of a Fault Alarm depends on the intentions of the operators and is therefore beyond the scope of this standard.

22.2.1 Creating Maintenance Domains and Maintenance Associations

The operator creates either a Default MD Level managed object or a Maintenance Domain managed object in a Bridge for every Maintenance Domain whose CFM PDUs can pass through that Bridge and are to be processed. The operator can also create a MA managed object in a Bridge for some or all of the MAs whose CFM PDUs can pass through that Bridge and are to be processed. Whether or not a VLAN can pass through a Bridge depends on:

a) Enable Ingress Filtering per-Bridge Port parameter (8.6.2);
b) Static VLAN Registration Entries (8.8.2);
c) Dynamic VLAN Registration Entries (8.8.5); and
d) Operation of the Multiple Spanning Tree protocol (Clause 13), which in turn, can be affected by the state of the MAC_Operational parameters (6.6.2) of the ISSs in the Bridge Port.

The first two of these methods are altered only by modifying managed objects and can therefore be considered static for the purposes of configuring CFM. The last two methods are dynamic, in that they depend not only on configuration, but on protocols that respond to network events, such as the addition or loss of a link or Bridge. Thus, every VLAN can pass through a Bridge unless there is a Static VLAN Registration Entry prohibiting that VLAN from passing through every Bridge Port, and Ingress Filtering is enabled on every Bridge Port. If a Bridge's Maintenance Domain requires only insignificant resources to support MHF configuration, then the most reliable network is one that assumes that all service instances will require MHFs at some MD Level higher than that of that Maintenance Domain.

22.2.2 Creating MEPs

There are two kinds of MA managed objects: those that are attached to one or more specific VIDs in the Bridge, and those that are attached to no VID.

Once the appropriate Maintenance Domain and MA have been created, a MEP can be created, modified, or deleted using the MA managed object defined in 12.14.6. MEPs are always created explicitly. MEP creation is controlled by creating a Maintenance association End Point managed object using the following parameters:

a) Whether the MA to which the MEP belongs is associated with specific VID(s), and if so, which VID(s) [item b) in 12.14.6.1.3], including a specification of the MA's Primary VID;
b) Whether the MEP is a Down MEP (pointing away from the MAC Relay Entity), or an Up MEP, (pointing towards the MAC Relay Entity) [item c) in 12.14.6.3.2].

Given those two values in the MEP managed object, Table 22-1 shows where, in Figure 22-9, the MEP is created. In Figure 22-9, the VID selects which of the MEP positions is selected for Table 22-1 column 3 values 1 and 4. Only one row of MEPs is shown in positions 1, 4, and 5, even though up to eight rows can exist at each of these points. Because the MEPs are always placed in order by MD Level, with the highest MD Level nearest the MHFs, at the center of each stack, there is thus no ambiguity.

Table 22-1—MEP creation

MA has VID [item b) in 12.14.6.1.3]	MEP direction [item c) in 12.14.7.1.3]	Position of MEP in Figure 22-9
Yes	Up	1
	Down	4
No	Up	Disallowed
	Down	5

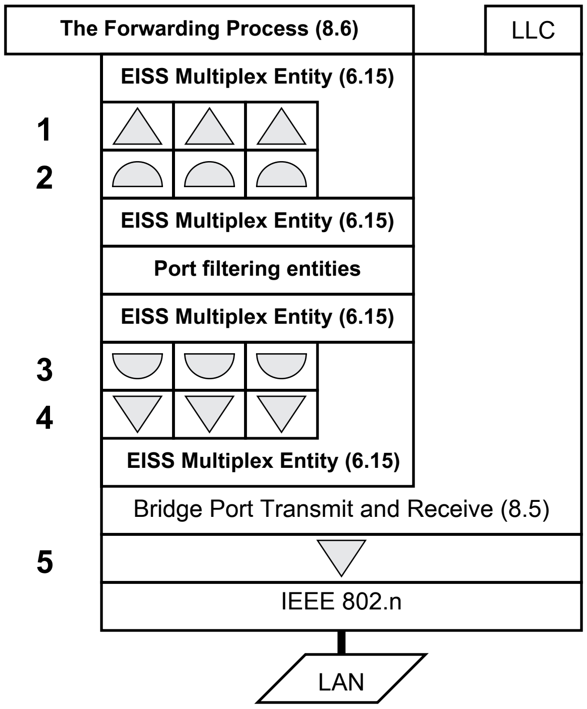
The Forwarding Process (8.6)
LLC
EISS Multiplex Entity (6.15)
1
2
EISS Multiplex Entity (6.15)
Port filtering entities
EISS Multiplex Entity (6.15)
3
4
EISS Multiplex Entity (6.15)
Bridge Port Transmit and Receive (8.5)
5
IEEE 802.n
LAN

Figure 22-9—Creating MEPs and MIPs

If any MEPs are configured on an MA, then in any given Bridge, either all of those MEPs are Up MEPs or all of them are Down MEPs; a Bridge refuses to create an Up (Down) MEPs if a Down (Up) MEP already is configured in that MA in that Bridge. An MA can be configured with Up MEPs in one Bridge and Down MEPs in another Bridge or Station. Any given VID can be configured on any number of MAs (including zero) containing either Down MEPs or no MEPs at all, plus zero or one MA containing Up MEPs. No VID can be configured on more than one MA on which are configured Up MEPs.

These restrictions, along with the configuration errors in 22.2.4, enable MEPs to be configured so that they bound an MA, no matter what action is taken by the spanning tree protocols within a Maintenance Domain.

22.2.3 Creating MIPs

Managed objects control the creation of MIPs, but indirectly, rather than explicitly, as for MEPs. Every MA defined in a Bridge can cause the management entity to create MIPs on every Bridge Port. On each Bridge Port, the following conditions trigger the management entity to reevaluate the creation of MIPs for a given VID (or list of VIDs) *x* on that Bridge Port:

a) The Bridge is initialized;
b) A MEP is created or deleted on VID(s) *x* on that Bridge Port;
c) An Up MEP is created or deleted on VID(s) *x* on any Bridge Port;
d) A change occurs in whether VID(s) *x* (any frame at all, for MAs not associated with any VID) can or cannot pass through the Bridge Port (22.2.1);
e) A change occurs in the Default MD Level managed object (12.14.3); or
f) A change occurs in a Maintenance Association managed object associated with VID(s) *x*.

A MIP is created by creating a pair of MHFs in positions 2 and 3 in Figure 22-9. Since MHFs cannot be created below the Bridge Port connectivity entity (8.5.1), MAs that are associated with no VID are not considered in this subclause. For a given VID on a given Bridge Port, and in the absence of any of the configuration errors in 22.2.4, there are at most two MAs at each MD Level that can affect MIP creation, plus at most one entry in the Default MD Level managed object. If one or more of the configuration errors in 22.2.4 are present, this standard does not specify what MIPs the Bridge creates.

When required, the Maintenance Entity performs the MIP evaluation on each Bridge Port for each VID. For each Bridge Port *p* and VID *x*, the Maintenance Entity creates a list of active MD Levels. This list contains:

g) MD Level of each of the MAs (if any) that includes VID *x* and a MEP configured on Bridge Port *p*;
h) MD Level of each of the MAs (if any) that includes VID *x* and has at least one Up MEP configured on some Bridge Port in this Bridge other than Port *p*;
i) MD Level of each of the MAs (if any) that includes VID *x* and has no MEPs configured on any Bridge Port in this Bridge; and
j) MD Level of the entry in the Default MD Level managed object (12.14.3) for VID *x*.

Exactly one MD Level *d* in this active list is eligible for MIP creation. It is the lowest MD Level in the list of active MD Levels such that there is no MEP configured on Bridge Port *p* and VID *x* at MD Level *d* or at any higher MD Level. This is the lowest MD Level in the list, if there is no such MEP configured. The Maintenance Entity then uses Table 22-2 to determine whether the MIP is to be created.

Table 22-2—MIP creation

Enumeration [item d) in 12.14.3.1.3, item c) in 12.14.5.1.3, or item c) in 12.14.6.1.3]	**A MEP is configured on Bridge Port *p***	**MIP created**
defMHFnone	—	No
defMHFexplicit	False	No
	True	Yes
defMHFdefault	—	Yes

The enumeration controlling whether the MIP is created is taken from:

k) The Maintenance Association managed object [item c) in 12.14.6.1.3] for the MA that includes VID *x*, and has at least one Up MEP configured on some Bridge Port in the Bridge [see item g)];
l) That MA's Maintenance Domain managed object [item c) in 12.14.5.1.3], if that MA exists, but its Maintenance Association managed object contains the value defMHFdefer; or
m) The Default MD Level managed object [item d) in 12.14.3.1.3] if no such MA exists [see item j)].

The enumerated values used in the managed objects item d) in 12.14.3.1.3, item c) in 12.14.5.1.3, and item c) in 12.14.6.1.3 to control the creation of a MIP for a VID on a Bridge Port are:

1) **defMHFnone:** No MHFs can be created for this VID(s) (the default value).
2) **defMHFdefault:** MHFs can be created for this VID(s) on any Bridge Port through which the VID(s) can pass where:
   i) There are no lower active MD levels; or
   ii) There is a MEP at the next lower active MD-level on the port.
3) **defMHFexplicit:** MHFs can be created for this VID(s) only on Bridge Ports through which this VID(s) can pass, and only if there is a MEP at the next lower active MD-level on the port.

4) **defMHFdefer:** In the Maintenance Association managed object [item c) in 12.14.6.1.3] only, the control of MHF creation is deferred to the corresponding variable in the enclosing Maintenance Domain [item c) in 12.14.5.1.3].

22.2.4 CFM configuration errors

While making the MIP creation evaluation described in 22.2.3, or whenever a MEP is created or deleted, the management entity can encounter errors in the configuration. Because of this, the Management entity maintains a Configuration Error List managed object (12.14.4) that lists, for every VID, the Bridge Ports that have MIP configuration errors. Every time the MIP creation evaluation is performed or a MEP is created or deleted, this list is updated to reflect which Bridge Ports and VIDs are in error and which are not. A Bridge Port is placed in the list if and only if one of the following errors occurs:

a) **CFMleak:** MA *x* is associated with a specific VID list, one or more of the VIDs in MA *x* can pass through the Bridge Port, no Up MEP is configured for MA *x* on the Bridge Port, no Down MEP is configured on any Bridge Port for MA *x*, and some other MA *y*, at a higher MD Level than MA *x*, and associated with at least one of the VID(s) also in MA *x*, does have a MEP configured on the Bridge Port;
b) **ConflictingVIDs:** MA *x* is associated with a specific VID list, an Up MEP is configured on MA *x* on the Bridge Port, and some other MA *y*, associated with at least one of the VID(s) also in MA *x*, and at the same MD Level as MA *x*, also has an Up MEP configured on some Bridge Port;
c) **ExcessiveLevels:** The number of different MD Levels at which MIPs are to be created on this port exceeds the Bridge's capabilities (see 22.3); or
d) **OverlappedLevels:** A MEP is created for one VID at one MD Level, but a MEP is configured on another VID at that MD Level or higher, exceeding the Bridge's capabilities (see 22.3).

A Bridge uses detection of the ExcessiveLevels error to refuse an operation on a managed object and thus avoids adding Bridge Ports to the error lists. This is because this incremental operation is known to be the one that exceeds the capabilities of the Bridge. However, a Bridge uses only the detection of the ExcessiveLevels error to refuse an operation on a managed object. That is because the other errors can arise as a result of a management operation, e.g., the deletion of a Static VLAN Registration Entry, that has no knowledge of CFM and cannot report the error. Furthermore, the root cause of either error might not be the last MA to be configured and could be under the control of another administration. Therefore, management operations that trigger errors other than ExcessiveLevels are performed, not rejected. The lists of Bridge Ports in error are maintained (12.14.4), so that the system administrator(s) can agree on a course of action, outside this standard, to correct the problem.

22.3 MPs, Ports, and MD Level assignment

It is possible to configure MPs at different MD Levels for different VLANs on a single Bridge Port. However, this imposes upon an implementation the burden to be able to configure up to 8189 MD Levels for the VLANs on a Port, 4094 for MHFs and 4095 for MEPs, each requiring 3 bits of storage. An implementation may not support different MD Levels for MIPs on different VIDs on a single Port, hence the ExcessiveLevels and OverlappedLevels error responses (22.2.4). At least one MD Level shall be supported for each port in a Bridge, such that MIPs can operate at that MD Level, and all CFM PDUs below that MD Level are filtered or processed, according to the configuration of MEPs on the port.

In Figure 18-7, the left-most Maintenance Entity at the lowest MD Level (0) has one MEP in the customer's equipment 1 and one in Provider Bridge 2. When Down MEPs are used in this manner to create a Maintenance Entity that interconnects two administrations, those administrations can agree upon a particular MD Level (and MAID and MEPIDs) to use. To enhance the chances for interoperability, the MD Level for Down MEPs should be the lowest not already in use by an existing MA, typically 0.

Assigning MD Levels with gaps between the levels used, as in Figure 18-7, can be wasteful of resources, especially Group registration entries in the Filtering Database. Minimizing the number of MD Levels, and concentrating the MD Levels used towards MD Level 0, minimizes the required resources, since CFM PDUs carrying higher MD Levels than those used in a Bridge are treated as ordinary multicasts by that Bridge. However, if one creates a Maintenance Domain at MD Level 0, and subsequently needs a level beneath that one, for example if a Provider Bridged Network is substituted for a physical connection, then the MD Level of the original Maintenance Domain might have to be changed. If there is another Maintenance Domain above the one at MD Level 0, that one would have to be changed, first, and so on. Similarly, if an existing Maintenance Domain at MD Level 3 is partitioned into several Domains, a new Maintenance Domain at MD Level 4 would be needed to ensure end-to-end connectivity across the extent of the old Maintenance Domain.

Such difficulties can be avoided in several ways:

a) The MD Levels of different Maintenance Domains can be insulated from each other by VLAN tags. Appropriate placement of the MEPs in Bridge Ports can ensure that the tags insulate the MD Levels, and that no reconfiguration will be necessary.
b) If it is expected that an additional MD Level will be needed because a physical link will likely become virtualized, MD Level 0 can be left vacant.
c) If it is expected that an additional MD Level will be needed because a Maintenance Domain will likely be partitioned, a gap can be left between that Maintenance Domain’s MD Level and the lowest MD Level offered to a customer of that Maintenance Domain.

See J.2 for a description of an MD Level assignment plan that can be used in the most general case.

22.4 Stations and Connectivity Fault Management

Stations, as well as Bridges, can implement Connectivity Fault Management (CFM). For the purposes of CFM, and for no other purposes, a Station is modeled as a Bridge, with the following exceptions:

a) This standard defines no entities above the layer of Down MPs in Figure 22-8. In particular, a Station has no Port filtering entities, Up MPs, or Forwarding Process. A CFM-capable Station is therefore a set of one or more disconnected Bridge Ports.
b) A Station shall not support the creation of MIPs or Up MEPs. Only Down MEPs can be implemented in a Station.
c) CFM PDUs not processed and not discarded by a MEP shall be ignored, and not processed, by the Station.
d) A Station shall create no entries in the Configuration Error List managed object (12.14.4) except for CFMleak errors.

22.5 Scalability of Connectivity Fault Management

CFM can create a considerable load on the resources of a Bridge. Ideally, a Bridge has the capability to reflect LBMs as LBRs at line rate, so that the capacity and reliability of a network can be tested. The total load imposed by LBMs, LBRs, LTMs, and LTRs cannot be predicted, as these are all under the control of the system administrator, and any number of streams of these PDUs can be generated.

The load imposed by CCMs is more regular, so reasonable predictions of the load they impose can be made. Based on a figure of 109 octets as the minimum frame size, from the start of the destination MAC address of one frame to the start of the destination MAC address of the next frame,[14] Table 22-3 shows the load that is imposed by CCM transmission and reception upon a Bridge Port with a single configured MEP, as a

[14]First TLV Offset = 70, 12 for MAC addresses, 6 for Type and fixed header, 1 for End TLV, 20 for inter-frame gap and preamble.

function of the CCM transmission interval, the number of MEPs in the MA, and the speed of the link. One row in the table shows the number of CCMs per second transmitted or received. The number in each of the other rows is the fraction of the total bandwidth required by those CCMs. Table 22-4 extends this calculation by multiplying all numbers by 1000, for the case where 1000 VLANs, each with a MEP, are configured on the Bridge Port, again with the specified transmission interval and average number of MEPs per MA.

Table 22-3—Bandwidth required for CCMs for 1 MA

Transmit interval	**3.3 ms**		**10 ms**		**1 s**		**60 s**	
MEPs in each MA	**10**	**2**	**10**	**2**	**10**	**2**	**10**	**2**
CCM / s	**3000**	**600**	**1 000**	**200**	**10**	**2**	**0.167**	**0.033**
10 Mb/s	26.160%	5.232%	8.720%	1.744%	0.087%	0.017%	0.001%	0.000%
100 Mb/s	2.616%	0.523%	0.872%	0.174%	0.009%	0.002%	0.000%	0.000%
1 Gb/s	0.262%	0.052%	0.087%	0.017%	0.001%	0.000%	0.000%	0.000%
10 Gb/s	0.026%	0.005%	0.009%	0.002%	0.000%	0.000%	0.000%	0.000%
100 Gb/s	0.003%	0.001%	0.001%	0.000%	0.000%	0.000%	0.000%	0.000%

Table 22-4—Bandwidth required for CCMs for 1000 MAs

Transmit interval	**3.3 ms**		**10 ms**		**1 s**		**60 s**	
MEPs in each MA	**10**	**2**	**10**	**2**	**10**	**2**	**10**	**2**
CCM / s	**3 000 000**	**600 000**	**1 000 000**	**200 000**	**10 000**	**2000**	**167**	**33**
10 Mb/s	> 100%	> 100%	> 100%	> 100%	87.200%	17.440%	1.453%	0.291%
100 Mb/s	> 100%	> 100%	> 100%	> 100%	8.720%	1.744%	0.145%	0.029%
1 Gb/s	> 100%	52.320%	87.200%	17.440%	0.872%	0.174%	0.015%	0.003%
10 Gb/s	26.160%	5.232%	8.720%	1.744%	0.087%	0.017%	0.001%	0.000%
100 Gb/s	2.616%	0.523%	0.872%	0.174%	0.009%	0.002%	0.000%	0.000%

Counting 4094 allowed VIDs in a VLAN tag, three kinds of MEPs, Up VID MEPs, Down VID MEPs, and Down no-VID MEPs, and eight MD Levels, a single Bridge Port with no Link Aggregation can, in theory, support 65 504 MEPs and 8188 MHFs.

22.6 CFM in Provider Bridges

The S-VLAN component of an IEEE 802.1ad Provider Bridge is, for the purposes of this standard, an example of a VLAN-aware Bridge. Fortunately, the many combinations possible when configuring CFM on Customer Bridges and Provider Bridges, connected using Port-based or S-tagged service interfaces, can be simplified.

22.6.1 Maintenance Points and C-VLAN components

When a C-VLAN component is integrated into a Provider Edge Bridge to form a C-tagged service interface, there is little justification for making all possible CFM entities manageable. IEEE Std 802.1ad-2005, 15.4, describes a C-tagged service interface, illustrated in Figure 15-4 of that standard. Figure 22-10 recasts IEEE 802.1ad Figure 15-4, illustrating the minimum set of CFM functions that a C-tagged service interface is required to support, if its Provider Bridge supports CFM. Almost all of the CFM components have been removed from the C-VLAN component. As a result, the transformations made in 22.1.2 and 22.1.3 are unnecessary, and the subclause references collapse to match those in IEEE 802.1ad Figure 15-4. The S-VLAN component contains almost the full suite of CFM entities. One set of shims has been moved from the Customer Network Port to the Customer Edge Port, however.

The managed objects in 12.14 support the creation of up to eight MEPs below the Bridge Port Transmit and Receive process (8.5). These MEPs belong to MAs associated with no VID. A MEP in this position would be of little use in a Customer Network Port, since there is no need to pair it with a MEP in the Provider Edge Port to protect an Internal LAN. However, a MEP attached to no VID in the Customer Edge Port could be used to protect the LAN to the customer equipment.

In order to support protection of the Customer Edge Port's LAN, while maximizing interoperability, whenever (Down) MEPs are created (12.14.6.3) in an MA associated with no VID, a Provider Edge Bridge shall allow those MEPs to be created using the interface number [item d) in 12.14.6.3.2] for a Customer Edge Port.

22.6.2 Maintenance C-VLAN on a Port-based service interface

Assume that a customer's C-VLAN Bridge is attached to an S-VLAN Bridge as in Figure 22-4, and that the customer wants to pair a Down MEP in the C-VLAN Bridge with a similar Down MEP in the C-VLAN Bridge on the other side of the Provider Bridged Network, thus creating an MA to provide end-to-end protection of the service instance offered by the Provider. In order to obtain the advantage of inserting the CFM PDUs above the Queuing entities, the customer would not use MEPs below the Bridge Port Transmit and Receive process (8.5). The customer could create one MA for each C-VID. However this would have drawbacks in that:

a) The extra MAs beyond the first MA would provide very little protection on a Port-based service;
b) Each extra MA would add to the overhead of CFM PDUs; and
c) The C-tagged CFM PDUs would not be detected by any of the CFM entities in any Provider Bridge, so no MIPs would be present in the customer's MAs.

If the customer, instead, creates an MA on only one C-VLAN, the Maintenance C-VLAN, and configures the two C-VLAN Bridges to pass the Maintenance C-VLAN across the Provider Bridge Network untagged, then:

d) The service instance is protected; and
e) The provider can configure a MIP on the S-VLAN carrying the customer's service instance that is visible in the customer's MA.

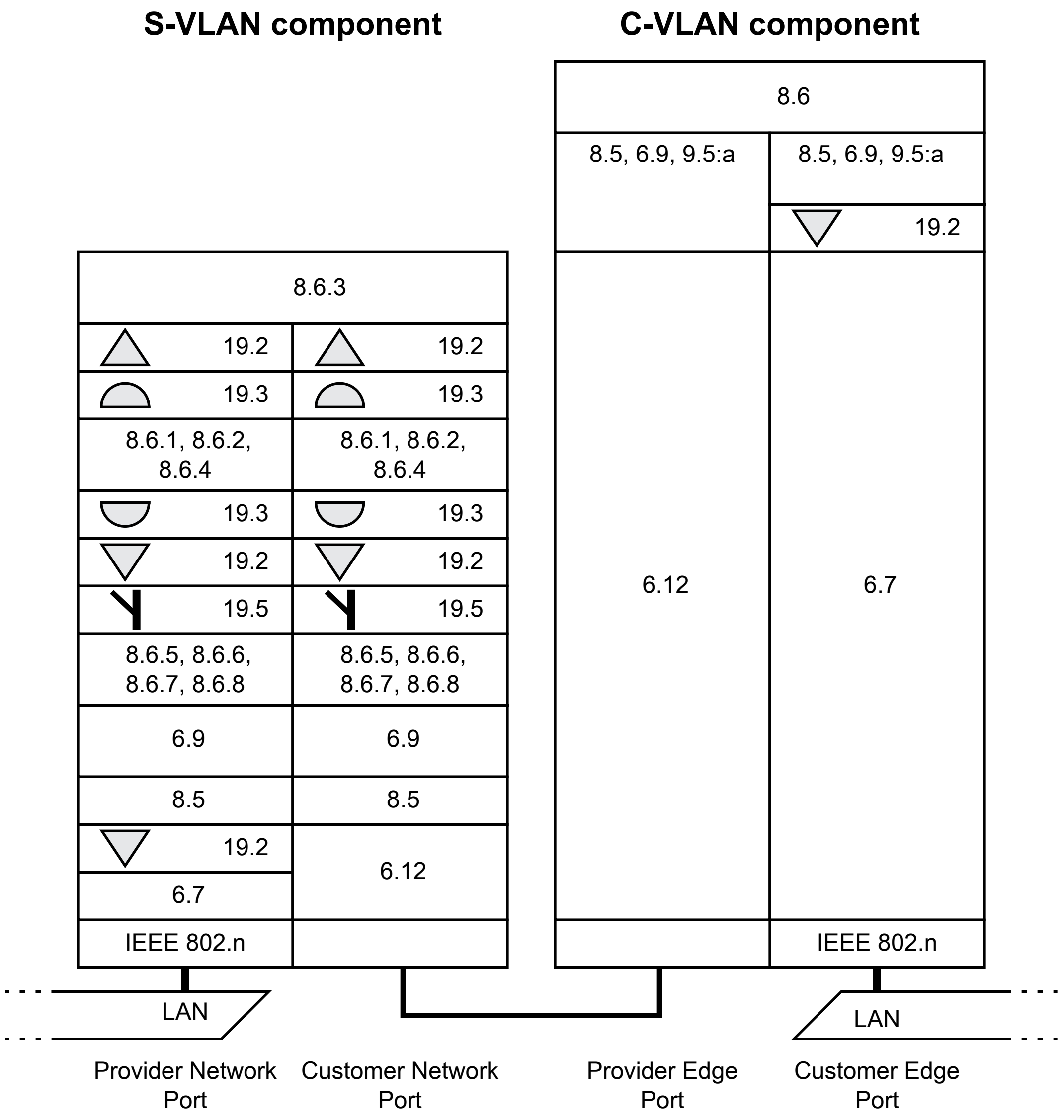

Figure 22-10—CFM in a Provider Edge Bridge C-tagged service interface

22.6.3 Maintenance C-VLAN on a C-tagged service interface

Assume that a customer's C-VLAN Bridge, illustrated by Figure 22-4 (less, of course, the box for "Support of the ISS for attachment to a Provider Bridged Network (6.11)"), is attached to a Provider Edge Bridge as in Figure 22-10. Assume that the customer has purchased several service instances, which have C-tagged service interfaces to other of the customer's C-VLAN Bridges at several different locations. Let us further suppose that the customer wants to pair Down MEPs in the C-VLAN Bridge to create MAs to provide end-to-end protection of the service instances. In order to obtain the advantage of inserting the CFM PDUs above the Queuing entities, the customer would not use MEPs below the Bridge Port Transmit and Receive process (8.5). The customer could create one MA for each C-VID.

There would be more justification in creating an MA per C-VLAN than for the Port-based service instance described in 22.6.2. However, if the customer is reasonably confident that the provider will carry each C-VLAN in the correct S-VLAN, then the customer can create multiple Maintenance C-VLANs, one per provider service instance (S-VLAN), and thus one per Provider Edge Port in the C-VLAN component. By configuring each of these Provider Edge Ports to pass that one C-VLAN untagged to the corresponding Customer Network Port in the S-VLAN component, the CFM PDUs in that one customer MA are visible to the S-VLAN component. The provider can configure a MIP for that S-VID that is visible to the customer's MA. The customer can have one MA protecting each service instance.

22.7 Management Port MEPs and CFM in the enterprise environment

Figure 22-11 illustrates the placement of Management Port MEPs. It is similar to the CFM Port illustrated in Figure J.1. However, the Management Port MEPs are configured explicitly for the Management Port. They do not represent information for any other Bridge Port in their CFM PDUs. Therefore, there is only one MEP for each MA. Furthermore, there can be no MHFs configured on a Management Port, because there can be no MEP further along the path to terminate the MA.

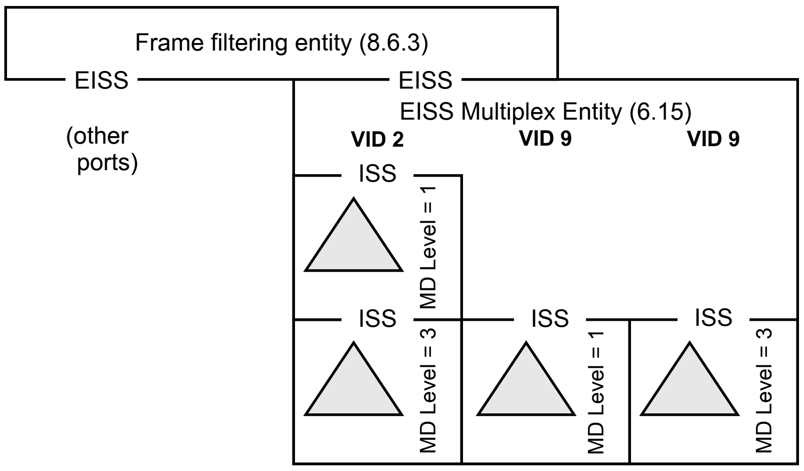

Figure 22-11—Up MEPs in a Management Port

NOTE—The fact that a CFM Port could be configured on a Management Port, and thus instantiate MHFs on the Management Port, does not contradict the preceding paragraph. Those MHFs are configured on a Bridge Port, and instantiated on the Management Port (CFM Port), not configured on the Management Port.

Although this standard uses the terminology of Provider Bridges and describes the functions of CFM in terms of Provider Bridges, it can be applied to any Bridge, whether a Provider Bridge, VLAN-aware Bridge, or non-VLAN-aware Bridge. The normative definitions and functions specified by this standard do not depend on, for example, whether the VLAN tag used to associate frames with VLANs use the customer format or the provider format. Certainly, no business relationships are required to implement this standard.

Given that both the users and the providers of an enterprise network are, by definition, all members of the same company, the utility of CFM in the enterprise environment is less than in the provider/customer environment. The network administrator of an enterprise network can find a number of uses for CFM, however.

The default values for MEP and MIP configuration make CFM easy to configure for the typical enterprise network. Figure 22-12 illustrates two Bridge Ports, the Management Port, and one Bridge Port connected to a LAN, using the default configuration as follows:

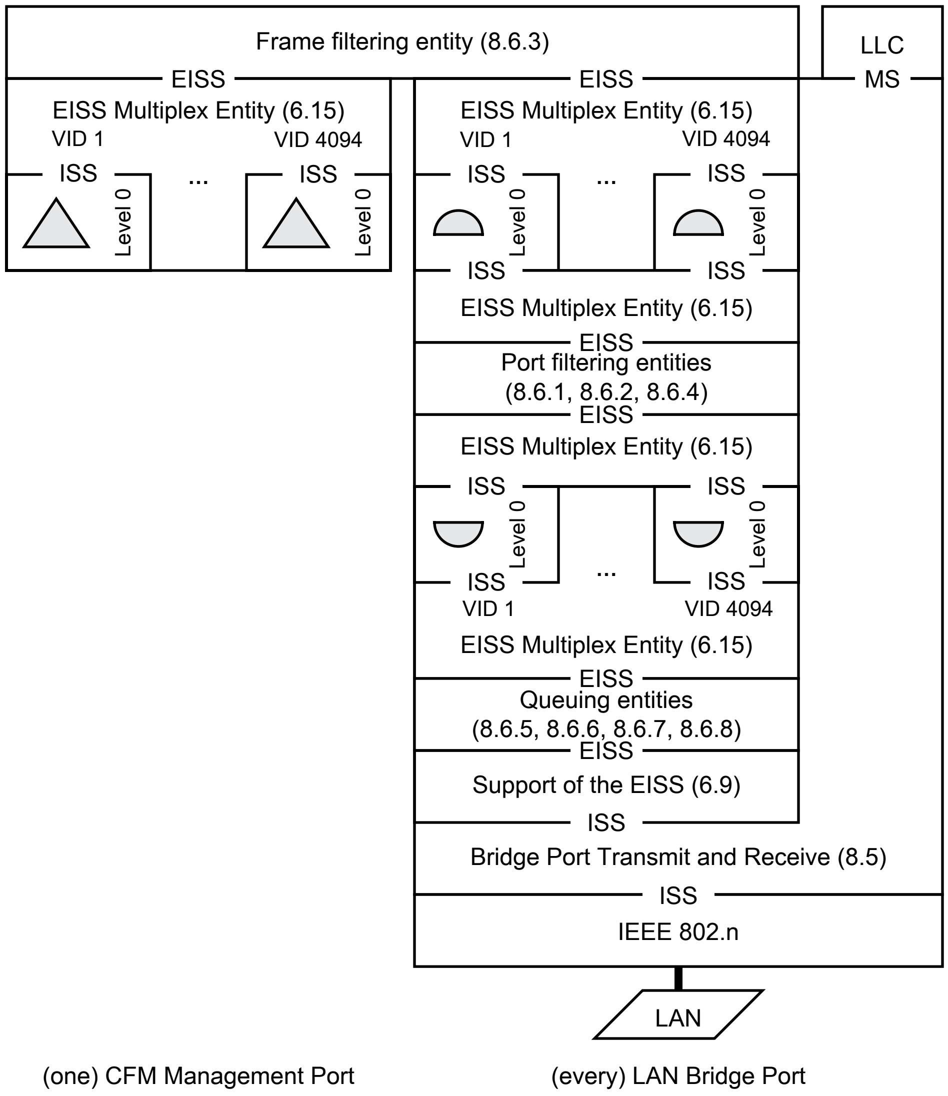

Figure 22-12—CFM in the enterprise environment

a) A single Maintenance Domain can be configured at MD Level 0.
b) An MA for each VLAN is configured in each Bridge.
c) A single MEP for each VLAN is configured on a Management Port in each Bridge. Typically, this is a port with no external interface, often the one used to manage the Bridge.
d) Through the Maintenance Domain managed object previously created, it is easy to define a MIP on every Bridge Port on every VLAN.

This simple configuration enables the administrator to use LBMs and LTMs to diagnose network faults among the Bridges of the network. In addition, specific high-priority VLANs, e.g., a VLAN interconnecting key routers and file servers for a large enterprise, could be explicitly monitored via CFM.

In a large enterprise, separate Maintenance Domains can be established, along with an accompanying enterprise-wide Maintenance Domain at a higher MD Level, in order to define and enforce separate spheres of responsibility for separate network administrations, e.g., in different departments in a university. The boundaries between Maintenance Domains might well coincide with administratively imposed boundaries between MSTP Regions.

22.8 Implementing CFM on existing Bridges

It might or might not be easy to implement CFM in a VLAN-aware Bridge conformant to IEEE Std 802.1Q-2005. The problems are best illustrated by a three-port Bridge as illustrated in Figure 22-13. Each of the three Bridge Ports A, B, and C is configured with MEPs and MIPs at the MD Levels shown in the figure.

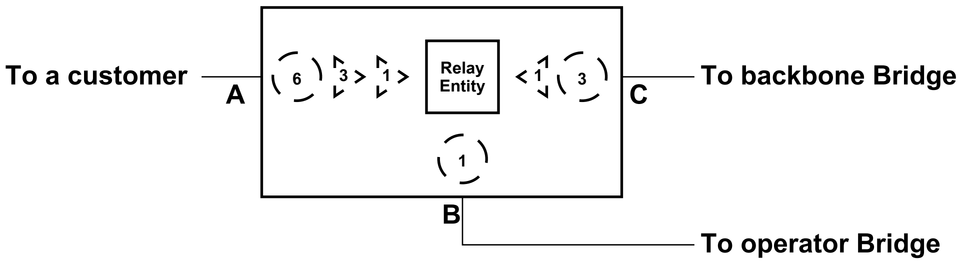

Figure 22-13—CFM on a Bridge conformant to IEEE Std 802.1Q-2005

If a CCM at MD Level 3 enters the Bridge from Bridge Port C, that CCM will be passed to both of the other Bridge Ports A and B. It will not exit Bridge Port A, however, because there is a MEP at MD Level 3 on that Bridge Port. It will, however, pass through Bridge Port B to the other operator Bridge. One can easily imagine configuring Group Registration Entry in the Bridge for the MD Level 3 group address shown in Table 8-9 that permits that group address to pass through the Bridge Ports B and C, but not through Bridge Port A, and also sends the CCM to a CFM module in the Bridge's Higher Layer Entities, perhaps residing on the Management Port of 22.7, for processing. It is precisely so that Group Registration Entries can be used in this manner that CCMs and LTMs are required to use the group addresses in Table 8-10 and Table 8-9.

A potential problem for some implementations can be seen if we ask what happens if that same CCM at MD Level 3 enters from Bridge Port A. In that case, the CCM does not exit either of Bridge Ports B or C, because Bridge Port A has a MEP at MD Level 3 to stop it; Group Registration Entries prevent frames from leaving a Bridge Port, not from entering one. Many Bridges have the ability to filter incoming frames based on the destination MAC address and/or the contents of the frame; either facility will enable this CCM to be stopped. In order to support both a Down MEP attached to no VID, and a Down MEP attached to a tagged VID at a lower MD Level than the no-VID MEP, the Bridge filters frames based on both the MAC address and whether the frame is tagged or not.

Annex A

(normative)

PICS proforma[15]

A.5 Major Capabilities

Insert the following items at the end of A.5:

Item	Feature	Status	References	Support
CFM	Is Connectivity Fault Management implemented?	O	5.4.1.3, 19, 20, 21, 22	Yes [ ] No [ ]
BRG	Is this system a Bridge, and not a Station, for the purposes of Connectivity Fault Management?	CFM: O	22.4	Yes [ ] No [ ]

A.7 Relay and filtering of frames

Insert the following items at the end of A.7:

Item	Feature	Status	References	Support
RLY-17	Are CFM frames discarded if they enter the queue subsequent to the Port leaving the Forwarding state?	CFM: X	item 8.6.7) in 8.6.7	No [ ]
RLY-18	Are CFM frames discarded when the Port leaves the Forwarding state?	CFM: O	8.6.7	Yes [ ] No [ ]

[15]*Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

A.14 Bridge Management

Insert the following items at the end of A.14:

Item	Feature	Status	References	Support
MGT-96	Does the Bridge provide control of all of the required CFM managed objects?	BRG AND CFM: M	item h) in 5.4.1.3, 12.14	Yes [ ]
	What method is used by the Bridge to provide control of all of the required CFM managed objects?			
MGT-97	The CFM Management Information Base (MIB) module defined in 17.5?	CFM: O.3	item n) in 5.4.1.3, 17.3, 17.5	Yes [ ] No [ ]
MGT-98	Some other method than the MIB module defined in 17.5?	CFM: O.3	17.3	Yes [ ] No [ ]
MGT-98a	If by some other method, what method is used?	MGT-101: M	17.3	____________
MGT-99	Does the Station provide control of all of the required CFM managed objects?	¬BRG AND CFM: M	item f) in 5.10, 12.14	Yes [ ]
	What method is used by the Station to provide control of all of the required CFM managed objects?			
MGT-100	The CFM Management Information Base (MIB) module defined in 17.5?	¬BRG AND CFM: O.4	item i) in 5.10, 17.3, 17.5	Yes [ ] No [ ]
MGT-101	Some other method than the MIB module defined in 17.5?	¬BRG AND CFM: O.4	17.3	Yes [ ] No [ ]
MGT-101a	If by some other method, what method is used?	MGT-101: M	17.3	____________
MGT-102	Is an entire C-tagged service interface given a single row in the IETF RFC 2863 IF-MIB?	PEB AND CFM: O	17.3	Yes [ ] No [ ]
MGT-103	Is every Bridge Port assigned its own conceptual row in the IETF RFC 2863 IF-MIB with its own unique ifIndex?	MGT-100: M	17.3	Yes [ ]
MGT-104	Does the Bridge support IEEE 802.3 Clause 43 Link Aggregation?	CFM: O	17.3, IEEE Std 802.3, Clause 43	Yes [ ] No [ ]
MGT-105	Does every IEEE 802.3 MAC, when aggregated via IEEE 802.3 Link Aggregation, have its own unique ifIndex, separate from the ifIndex of the Bridge Port as a whole?	MGT-104: M	17.3	Yes [ ] No [ ]
	Questions MGT-106 through MGT-131 refer to operations related to CFM managed objects.			
MGT-106	Read Maintenance Domain list	CFM: M	12.14.1.1	Yes [ ]
MGT-107	Create Maintenance Domain managed object	CFM: M	12.14.1.2	Yes [ ]
MGT-108	Delete Maintenance Domain managed object	CFM: M	12.14.1.3	Yes [ ]
MGT-109	Read CFM Stack managed object	CFM: M	12.14.2.1	Yes [ ]
MGT-110	Read Default MD Level managed object	CFM: M	12.14.3.1	Yes [ ]
MGT-111	Write Default MD Level managed object	CFM: M	12.14.3.2	Yes [ ]
MGT-112	Read Configuration Error List managed object	CFM: M	12.14.4.1	Yes [ ]
MGT-113	Read Maintenance Domain managed object	CFM: M	12.14.5.1	Yes [ ]
MGT-114	Write Maintenance Domain managed object	CFM: M	12.14.5.2	Yes [ ]
MGT-115	Create Maintenance Association managed object	CFM: M	12.14.5.3	Yes [ ]

Item	Feature	Status	References	Support
MGT-116	Delete Maintenance Association managed object	CFM: M	12.14.5.4	Yes [ ]
MGT-117	Read Maintenance Association managed object	CFM: M	12.14.6.1	Yes [ ]
MGT-118	Write Maintenance Association managed object	CFM: M	12.14.6.2	Yes [ ]
MGT-119	Create Maintenance association End Point managed object	CFM: M	12.14.6.3	Yes [ ]
MGT-120	Delete Maintenance association End Point managed object	CFM: M	12.14.6.4	Yes [ ]
MGT-121	Read Maintenance association End Point managed object	CFM: M	12.14.7.1	Yes [ ]
MGT-122	Total number of out-of-sequence CCMs received	CFM: O	item v) in 12.14.7.1.3, 20.16.12	Yes [ ] No [ ]
MGT-123	Total number of LBRs received with data match errors	CFM: O	item aa) in 12.14.7.1.3	Yes [ ] No [ ]
MGT-124	Write Maintenance association End Point managed object	CFM: M	12.14.7.2	Yes [ ]
MGT-125	Transmit Loopback Messages	CFM: M	12.14.7.3	Yes [ ]
MGT-126	Transmit Linktrace Message	CFM: M	12.14.7.4	Yes [ ]
MGT-127	Read Linktrace Reply	CFM: M	12.14.7.5	Yes [ ]
MGT-128	Read MEP Database	CFM: M	12.14.7.6	Yes [ ]
MGT-129	Read received Port Status TLV	CFM: O	item f) in 12.14.7.6.3, 20.19.3	Yes [ ] No [ ]
MGT-130	Read received Interface Status TLV	CFM: O	item g) in 12.14.7.6.3, 20.19.4	Yes [ ] No [ ]
MGT-131	Transmit MEP Fault Alarm	CFM: M	12.14.7.7	Yes [ ]

A.23 Connectivity Fault Management

Insert the following subclause after A.22:

Item	Feature	Status	References	Support
CFM-1	Does the Bridge support the creation of Maintenance Domains at eight MD Levels, with multiple Maintenance Domains at each MD Level?	BRG AND CFM: M	item a) in 5.4.1.3	Yes [ ]
CFM-2	Does the Bridge support the creation of a Maintenance Association (MA) on each VLAN supported by the Bridge for each MD Level?	BRG AND CFM: M	item b) in 5.4.1.3	Yes [ ]
CFM-3	Does the Bridge support the creation of a single MIP for each Maintenance Domain on each Port, all MIPs being at the same MD Level?	BRG AND CFM: M	item c) in 5.4.1.3	Yes [ ]
CFM-4	Does the Bridge support the creation of eight Up MEPs on each VLAN on each Port, each MEP at a different MD Level?	BRG AND CFM: M	item d) in 5.4.1.3	Yes [ ]
CFM-5	Does the Bridge support the creation of eight Down MEPs on each VLAN on each Port, each MEP at a different MD Level?	BRG AND CFM: M	item e) in 5.4.1.3	Yes [ ]

Item	Feature	Status	References	Support
CFM-6	Does the Bridge support the creation of eight Down MEPs associated with no VLAN on each Port, each MEP at a different MD Level?	BRG AND CFM: M	item f) in 5.4.1.3	Yes [ ]
CFM-7	Does the Bridge support the maintenance of a MEP CCM Database?	BRG AND CFM: M	item g) in 5.4.1.3	Yes [ ]
CFM-8	Does the Bridge conform to the state machines and procedures in Clause 20?	BRG AND CFM: M	item i) in 5.4.1.3, 20	Yes [ ]
CFM-9	Does the Bridge transmit and accept frames in the formats specified in Clause 21?	BRG AND CFM: M	item j) in 5.4.1.3	Yes [ ]
CFM-10	Does the Bridge support the creation of MIPs at different MD Levels on a single Port?	BRG AND CFM: O	item k) in 5.4.1.3, 22.3	Yes [ ] No [ ]
CFM-11	Does the Bridge support the creation of MEPs at MD Levels equal to or higher than the MD Levels of MIPs on other VIDs on the same Port?	BRG AND CFM: O	item l) in 5.4.1.3, 22.3	Yes [ ] No [ ]
CFM-12	Does the Bridge support the maintenance of a MIP CCM Database in MIPs and MEPs?	BRG AND CFM: O	item m) in 5.4.1.3, 19.2.8, 19.3	Yes [ ] No [ ]
CFM-13	Does the Bridge support the creation of MAs that are associated with more than one VLAN?	BRG AND CFM: O	item o) in 5.4.1.3	Yes [ ] No [ ]
CFM-14	Does the Station support the creation of Maintenance Domains at eight MD Levels, with multiple Maintenance Domains at each MD Level?	¬BRG AND CFM: M	item a) in 5.10	Yes [ ]
CFM-15	Does the Station support the creation of a Maintenance Association (MA) on each VLAN supported by the Bridge for each MD Level?	¬BRG AND CFM: M	item b) in 5.10	Yes [ ]
CFM-16	Does the Station support the creation of MIPs?	¬BRG AND CFM: X	item b) in 22.4	No [ ]
CFM-17	Does the Station support the creation of Up MEPs?	¬BRG AND CFM: X	item b) in 22.4	No [ ]
CFM-18	Does the Station support the creation of eight Down MEPs on each VLAN on each Port, each MEP at a different MD Level?	¬BRG AND CFM: M	item c) in 5.10	Yes [ ]
CFM-19	Does the Station support the creation of eight Down MEPs associated with no VLAN on each Port, each MEP at a different MD Level?	¬BRG AND CFM: M	item d) in 5.10	Yes [ ]
CFM-20	Does the Station support the maintenance of a MEP CCM Database?	¬BRG AND CFM: M	item e) in 5.10	Yes [ ]
CFM-21	Does the Station conform to the state machines and procedures in Clause 20?	¬BRG AND CFM: M	item g) in 5.10, 20	Yes [ ]
CFM-22	Does the Station transmit and accept frames in the formats specified in Clause 21?	¬BRG AND CFM: M	item h) in 5.10	Yes [ ]
CFM-23	Does the Station support the creation of MAs that are associated with more than one VLAN?	¬BRG AND CFM: O	item j) in 5.10	Yes [ ] No [ ]
CFM-24	Does the MP Level Demultiplexer discard frames that are too short to contain an MD Level header field?	CFM: M	19.2.6, 20.46.4.1	Yes [ ]
CFM-25	Is an LBM always discarded, and not replied to, if its source_address is a Group address?	CFM: M	20.2.2	Yes [ ]
CFM-26	Are the contents of an LBM, except for the source_address and OpCode, ignored and not interpreted by the receiver?	CFM: M	20.2.2	Yes [ ]

Item	Feature	Status	References	Support
CFM-27	Is the LTFwhile timer implemented?	CFM:M	20.5, 20.5.1	Yes [ ]
CFM-28	Are the per-MEP timers CCIwhile, errorCCM-while, xconCCMwhile, LBIwhile, and FNGwhile implemented?	CFM:M	20.5, 20.5.2, 20.5.3, 20.5.4, 20.5.5, 20.5.6	Yes [ ]
CFM-29	Is the per-MEP per-remote MEP timer rMEPwhile implemented?	CFM:M	20.5, 20.5.7	Yes [ ]
CFM-30	Can a MEP set rMEPCCMdefect within (3.25 * CCMtime(CCMinterval)) seconds of the receipt of a CCM?	X	20.5.7	No [ ]
CFM-31	Can a MEP take longer to set rMEPCCMdefect than (3.5 * CCMtime(CCMinterval)) seconds of the receipt of the last CCM?	CFM: M	20.5.7	Yes [ ]
CFM-32	Does the system transmit a non-0 value for the CCM Interval in CCMs?	CFM: M	20.8.1, item f) in 20.11.1	Yes [ ]
CFM-33	Does the Bridge transmit all CFM PDU fixed header fields in conformance with this standard?	CFM: M	20.46.2	Yes [ ]
CFM-34	Does the Bridge transmit all reserved bits and fields in CFM PDUs as 0?	CFM: M	20.46.2	Yes [ ]
CFM-35	Does the Bridge transmit additional fixed header fields not defined in this standard in CFM PDUs?	X	20.46.2	No [ ]
CFM-36	Does the Bridge transmit code points in any field that are reserved, either by this standard or by ITU-T Y.1731 (2006)?	X	20.46.2	No [ ]
CFM-37	Does the Bridge transmit additional fields in any CFM PDU in any TLV defined by this standard?	X	20.46.2	No [ ]
CFM-38	Does the Bridge determine the validity of those CFM PDUs that are validated, in a manner indistinguishable, by external observation of the Bridge, from the procedures described in this standard?	CFM: M	20.46.3	Yes [ ]
CFM-39	Is the version by which each PDU is processed selected correctly, and are the prohibited validation criteria not applied, by the System?	CFM: M	20.17.1, 20.17.2, 20.26.1, 20.31.1, 20.46.4.2	Yes [ ]
CFM-40	Does the System determine the validity of a CFM PDU in the manner defined by this standard?	CFM: M	20.17.1, 20.17.2, 20.26.1, 20.31.1, 20.46.4.1, 20.46.4.3	Yes [ ]
CFM-41	Are all CFM PDUs transmitted with an integral number of octets?	CFM: M	21.1	Yes [ ]
CFM-42	Does the Bridge transmit Organization-Specific TLVs?	CFM: O	20.46.2, 21.5.2	Yes [ ] No [ ]
CFM-43	Does the Bridge transmit an Organization-Specific TLV that requires it or the receiver to violate any requirement of this standard?	X	21.5.2	No [ ]
CFM-44	Is the information transmitted in an Organization-Specific TLV independent from information in a TLV received from any other port?	CFM-42: M	21.5.2	Yes [ ]
CFM-45	Do Organization-Specific TLV(s) transmitted by the Bridge provide a means for sending messages that are larger than would fit within a single CFM PDU?	X	21.5.2	No [ ]

Item	Feature	Status	References	Support
CFM-46	Do the Organization-Specific TLV(s) transmitted by the Bridge conform to the validation and versioning rules of 20.46?	CFM-42: M	21.5.2	Yes [ ]
CFM-47	Does the Management Address Domain field in the Sender ID TLV(s) transmitted by the Bridge (if any) conform to ITU-T X.690 8.19?	CFM: M	21.5.3.5	Yes [ ]
CFM-48	Can a MEP in an multiple spanning tree environment not statically restricted to a single MSTI generate a Port Status TLV?	X	21.5.4	No [ ]
CFM-49	Does the Bridge, in any case except when receiving an LBR, interpret the contents of the Data TLV?	X	21.5.6	No [ ]
CFM-50	Can the Bridge transmit the Data TLV in an LBM?	CFM: O	21.5.6	Yes [ ] No [ ]
CFM-51	Is the Bridge able to receive and process all valid CCM PDUs that are 128 octets or less in length (from MD Level through End TLV)?	CFM: M	21.6	Yes [ ]
CFM-52	Does the Bridge discard, as invalid, CCM PDUs that exceed 128 octets in length?	CFM: O	21.6	Yes [ ] No [ ]
CFM-53	Can the Bridge transmit a CCM PDU that is longer than 128 octets in length?	X	21.6	No [ ]
CFM-54	Is the length of the MAID transmitted in a CCM exactly 48 octets?	CFM: M	21.6.5	Yes [ ]
CFM-55	Is the field Defined by ITU-T Y.1731 (2006) always transmitted as 0?	CFM: M	21.6.6	Yes [ ] N/A [ ]
CFM-56	Does the First TLV Offset field contain the value as specified for the OpCode for each CFM message transmitted?	CFM: M	21.4.5, 21.6.2, 21.7.2, 21.8.2, 21.9.2	Yes [ ]
CFM-57	Does every transmitted LTM contain an LTM Egress Identifier TLV?	CFM: M	21.8.7	Yes [ ]
CFM-58	Does every transmitted LTR contain an LTR Egress Identifier TLV?	CFM: M	21.9.6	Yes [ ]
CFM-59	Does the receiving Bridge behave differently if the order of TLVs in a CFM PDU, other than the End TLV, or TLVs in an LBR, is altered?	X	21.6.7, 21.9.6	No [ ]
CFM-60	Does the Bridge support the creation of MPs at one or more MD Level on every Port?	CFM: M	22.3	Yes [ ]
CFM-61	Is every Down MEP on a Bridge Port assigned a MAC address different than any Down MEP on any other Bridge Port?	CFM: M	19.4	Yes [ ]
CFM-62	Does the Provider Edge Bridge support the creation of a Down MEP on the interface corresponding to a Customer Edge Port?	PEB AND BRG AND CFM: M	22.6.1	Yes [ ]
CFM-63	If the MIP CCM Database has insufficient resources to record a new entry, does it preferentially remove the oldest entry to make room for the new one?	BRG AND CFM-12: O	20.1.3	Yes [ ] N/A [ ]
CFM-64	Does the Bridge transmit successive integer values in the Sequence Number field of a CCM?	CFM: O	20.1, item h) in 20.11.1	Yes [ ] N/A [ ]
CFM-65	Does the Bridge transmit neither successive integer values nor 0 in the Sequence Number field of a CCM?	X	20.1, 20.11.1	No [ ]

Item	Feature	Status	References	Support
CFM-66	Does the Bridge keep track of received CCMs' Sequence Number fields in the MEP CCM Database, and count out-of-sequence CCMs in CCMsequenceErrors?	CFM: O	20.1, 20.17.1	Yes [ ] N/A [ ]
CFM-67	Does the Bridge check the validity of every received LTM, and LTR?	CFM: M	20.31.1, 20.39.1, 20.42.1	Yes [ ]
CFM-68	Does the Bridge check the validity of every received CCM?	CFM: O	20.17.1, 20.17.2	Yes [ ] No [ ]
CFM-69	Does the Bridge check the validity of every received LBM?	CFM: O	20.2.2, 20.26.1	Yes [ ] No [ ]
CFM-70	Does the Bridge check the validity of every received LBR?	CFM: O	20.2.2, 20.31.1	Yes [ ] No [ ]
CFM-71	Does the Bridge compare the received LBR bit-by-bit against the original LBM?	CFM: O	20.2.3, 20.31.1	Yes [ ] No [ ]
CFM-72	Can the Bridge transmit LBMs at a rate fast enough to overflow output queues in the absence of other data traffic?	X	20.2.1	No [ ]
CFM-73	Can a high rate of incoming CFM PDUs increase the probability that the Bridge fails to protect the network against forwarding loops?	BRG: X	20.46.5	No [ ]
CFM-74	Are the Optional CCM TLVs included in every CCM?	CFM: O	21.6.7	Yes [ ] N/A [ ]
CFM-75	Are Organization-Specific TLVs included in CCMs?	CFM: O	21.6.7	Yes [ ] No [ ]
CFM-76	Is the Sender ID TLV included in every LBM?	CFM:O	21.5.3, 21.7.4	Yes [ ] N/A [ ]
CFM-77	Is the Management Address included in the Sender ID TLV?	CFM-76: O	21.5.3	Yes [ ] No [ ]
CFM-78	Are Organization-Specific TLVs included in LBMs?	CFM: O	21.7.4	Yes [ ] No [ ]
CFM-79	Does the Station discard, and not process, all CFM PDUs not discarded or processed by its MEPs?	¬BRG AND CFM: M	item c) in 22.4	Yes [ ]
CFM-80	Does the Station create entries in the Configuration Error List managed object other than CFMleak errors?	X	item d) in 22.4, 12.14.4	No [ ]

Annex H

(informative)

Bibliography

Insert the following items in alphanumeric order, renumber the existing entries, and update globally:

[B1] Asynchronous Transfer Mode (ATM): A collection of equipment and standards used for telecommunications and data transfer. (See http://www.itu.int/ITU-T/ and http://www.atmforum.com.)

[B2] IEEE Std 802a-2003, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture—Amendment 1: Ethertypes for Protoypes and Vendor-specific Protocol Development.

[B3] IEEE Std 802.1AB-2005, IEEE Standard for Local and Metropolitan Area Networks—Station and Media Access Control Connectivity Discovery.

[B4] IEEE Std 802.1AE-2006, IEEE Std for Media Access Control (MAC) Security.

[B5] IEEE Std 802.11-2007, Standard for LAN/MAN—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.

[B6] IETF RFC 3031 (Proposed standard), Multiprotocol Label Switching Architecture; The Internet Society; http://www.ietf.org/rfc/rfc3031.txt.

[B7] IETF RFC 3410 (Informational), Introduction and Applicability Statements for Internet Standard Management Framework.

[B8] ITU-T G.8010 (2003), Transmission Systems and Media, Digital Systems and Networks Digital Networks.

[B9] ITU-T G.8011 (2004), Ethernet Over Transport—Ethernet Services Framework Digital Networks.

[B10] ITU-T G.8011.1 (2004), Ethernet Private Line Service Digital Networks.

[B11] ITU-T G.8012 (2004), Ethernet UNI and Ethernet NNI: Transmission Systems and Media, Digital Systems and Networks Digital Networks.

[B12] ITU-T G.8021 (2004), Characteristics of Ethernet Transport Network Equipment Functional Blocks: Transmission Systems and Media, Digital Systems and Networks Digital Networks.

[B13] ITU-T I.610 (1999), B-ISDN operation and maintenance principles and functions; http://www.itu.int/publications/.

[B14] ITU-T Q.5/13, Question 5 (OAM and network management for NGN) of ITU-T Study Group 13 (Study Period 2005–2008).

[B15] ITU-T X.25 (1996), Public Data Networks: Interfaces—Interfaces between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit.

[B16] Ethernet Local Management Interface (E-LMI), Metro Ethernet Forum Technical Specification MEF 16, 2006; http://www.metroethernetforum.org/PDFs/Standards/MEF16.pdf.

[B17] Multiprotocol Label Switching (MPLS): A standard for label-based forwarding in an IP network. The standard is specified in several RFCs (see http://www.ietf.org/html.charters/mpls-charter.html) and ITU-T recommendations (see http://www.itu.int/ITU-T/).

Insert a new annex preceding the Bibliography, and renumber the annexes so that the Bibliography is the last annex in the standard.

Annex J

(informative)

Connectivity Fault Management protocol design and use

J.1 Origin of Connectivity Fault Management

Operations, Administration, and Maintenance functions, or OAM, are a product of the world of telephony. As Ethernet, which has traditionally been an Enterprise network technology, expands into the realm of service providers, such useful notions as OAM become important. The OAM work going on in various standards bodies and in various vendors' products can be classified into five groups:

1) Provider Edge OAM, as defined by IEEE 802.3, Clause 57.
2) Ethernet Local Management Interface (E-LMI) [B16], as defined by the Metro Ethernet Forum.
3) Underlying layers of OAM, such as MPLS OAM or ATM OAM,[16] used on various sorts of emulated Ethernet links, carrying Ethernet frames as a higher layer.
4) Ethernet frames carrying End-to-End Connectivity Fault Management (CFM) PDUs, defined by this standard and by ITU-T Q.5/13 [B14].
5) Network performance measurement, being defined by the Metro Ethernet Forum and ITU-T, including Recommendation ITU-T Y.1731 (2006).

Connectivity Fault Management (CFM) is one part of the whole OAM picture. It defines category 4, Ethernet frames carrying End-to-End CFM PDUs, defined by this standard and by ITU-T Q.5/13. This standard does not explicitly address the subject of the configuration or provisioning of Metro Ethernet services, but it does generate significant requirements on configuration and provisioning. Certain CFM PDU formats are also useful for network performance measurement (item 5) and are defined in this standard. Additional PDUs for these operations are defined in ITU-T Y.1731.

Finally, unlike OAM in the telephony world, CFM does not address recovery from a loss of connectivity. That is the function of the various Layer 2 control protocols such as spanning tree (see Clause 13).

J.2 Deployment of Connectivity Fault Management

It is critical to the early deployment of CFM that no hardware changes be required to achieve a significant level of CFM functionality. However, if the MEPs in a large number of large Maintenance Associations issue Continuity Check Messages (CCMs) at the highest allowed rates, a Provider Bridge's ability to generate and/or absorb these CCMs could be overwhelmed. To achieve its full potential, CFM could require hardware modifications to existing Provider Bridges.

a) The CFM shims shown in Figure 22-4 are an abstract concept. They have been defined so that it should be possible to emulate their functions on most existing platforms, though perhaps not at high rates of CFM PDU transmission and reception. Hardware assistance can enable higher CFM PDU

[16]For references to these technologies, see the Bibliography in Annex H.

rates and can enable the implementation of the Shared MP address model of operation instead of the Individual MP address model (J.6).

b) Directing CFM PDUs to the appropriate ports of a Provider Bridge could necessitate a separate Group Registration Entry for each VID in the filtering database. Some means whereby this large number of entries could be abstracted would reduce the load on the filtering database and enable the use of a larger number of MD Levels.

c) The reception of a Loopback Message and generation of the corresponding Loopback Reply have been defined in a manner that is amenable to implementation in hardware. If so implemented, then the providers' ability to support a particular requirement for bandwidth in a service instance can easily be tested.

J.3 MD Level allocation alternative

There are eight Maintenance Domain (MD) Levels as shown in Table J.1. In order to allow administrations to configure different MDs in the same Bridge without having to resort to detailed inter-organizational agreements as to which MD Levels are available for use, an administrator of an MD can decide which of three roles that MD will play: the "customer" role, the "service provider" role, or the "operator" role. The administrator would then be free to assign an MD Level to an MD within the range shown in Table J.1 according to the MD's role. If two administrations can operate MDs using the same role in one Bridge at different MD Levels.

Table J.1—Provider MD Level allocation

MD Level	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Use	customer			service provider		operator		
	Highest	<- Higher lower ->						Lowest

This method of MD Level allocation attempts to minimize the likelihood of incompatibilities caused by MD Level selection among separate administrative organizations that interconnect in a manner such that their MD Levels are visible to each other. Due to the addition of encapsulations such as VLAN tags that can hide one organization's MD Levels from another, it is considered unlikely that the allocation method shown in Table J.1 will be necessary and that the method described in 22.3 will be sufficient.

J.4 Relationship of IEEE Std 802.1ag-2007 to other standards

ITU-T Y.1731 (2006) is a compatible extension of CFM. Certain terminology is different between ITU-T Y.1731 and this standard. These differences are summarized in Table J.2.

ITU-T Y.1731 includes a number of OpCodes in addition to the CCM, LBM, LBR, LTM, and LTR defined in this standard. ITU-T Y.1731 also specifies that the LBM can have a Group destination_address, in addition to the ability of using an Individual destination_address, as defined in this standard. It is for this reason that the MP Loopback Responder's ProcessLBM() procedure (20.26.1) is required to respond to an LBM whose destination_address is the appropriate CCM Group address, even though this standard specifies no means for generating such a frame.

Table J.2—IEEE / ITU-T terminology differences

IEEE Std 802.1ag-2007	ITU-T Y.1731 (2006)
Maintenance Association (MA)	Maintenance Entity Group (MEG)
Maintenance Association Identifier (MAID)	Maintenance Entity Group Identifier (MEGID)
Maintenance Domain (MD)	(No such construct present)
Maintenance Domain Level (MD Level)	Maintenance Entity Group Level (MEG Level)

The MIP is defined as two MHFs in this standard, but not in ITU-T Y.1731. This is because ITU-T Y.1731 is not concerned with the details of the implementation.

ITU-T Y.1731 does not include an LTM Egress Identifier TLV (21.8.8) or LTR Egress Identifier TLV (21.9.7). For compatibility with early implementations of ITU-T Y.1731, this standard does not require that these TLVs be present on received LTMs or LTRs. However, this standard does require these TLVs to be transmitted in LTMs and LTRs.

ITU-T Y.1731 specifies that, if an LTM passes through MPs on both the ingress and egress ports, the TTL field of the LTM will be decremented twice. This standard requires, in that case, that the Bridge decrement the TTL field only once, and return a single LTR. The Y.1731 behavior is compatible with the LTR analysis presented in J.5, as long as each of the MPs that decrement the LTM's TTL field also return an LTR. If a Bridge decremented the TTL twice, but returned only a single LTR, it would be impossible for the originating MEP to tell whether or not an LTR was lost in transit.

The Flags field (21.9.1) returned in the LTR contains two extra bits of information in this standard that are not in ITU-T Y.1731.

J.5 Interpreting Linktrace results

The LTR entries are retrieved from the Linktrace Database in the order the LTRs were received by the MEP. There is no way to determine when the last LTR has been received, except to wait for some number of seconds (e.g., long enough for a worst-case delay through the Bridged Network plus a few seconds), after the successful completion of the Transmit Linktrace Message command, before assuming that no further LTRs are likely to be received.

Because of the timer used by the LTR Transmitter state machine (20.5.1), the order in which the LTRs are received (time order) is likely to be different from the order of the MPs reached by the LTM as it was forwarded, MP by MP (path order). It is possible for an LTM to take more than one path, so that the LTRs report a tree, rooted at the originating MEP, rather than a simple linear series of zero or more MHFs terminating at a MEP or an MHF. If there are no network topology changes during the Linktrace operation, and if all systems' behavior conforms to this standard, the following can be said about constructing this path tree from the list of LTR entries returned by the Read Linktrace Reply command:

a) Sort the returned LTR entries by decreasing value of ltrReplyTTL (20.36.2.2). The highest reported value is one less than the initial LTM TTL field value [item d) in 12.14.7.4.2] used in the Transmit Linktrace Message command, and corresponds to the first Linktrace Responder reached by the initial LTM. This Linktrace Responder resides in the same Bridge as the MEP, in the case of an LTM originated by an Up MEP. The next-lower value corresponds to the MHF(s) or MEP(s) reached by the first forwarded copy of that LTM, and so on. Any gaps in the sequence of Reply TTL values indicate lost LTRs.

b) The ltrNextEgressId variable (20.36.2.4) in every LTR entry returned from a given Transmit Linktrace Message command is unique among the LTR entries in a given LTM entry in the Linktrace database, since no Linktrace Responder transmits more than one copy of an LTM.
c) The ltrLastEgressId variable (20.36.2.3) in any LTR entry is unique only over those LTR entries with matching values of the ltrReplyTTL variable.
d) The ltrLastEgressId variable (20.36.2.3) in any LTR entry whose ltrReplyTTL variable is one less than the initial LTM TTL field also matches the original LTM's LTM Egress Identifier TLV value, as returned from the Transmit Linktrace Message command [item c) in 12.14.7.4.3].
e) In the absence of lost LTRs, the ltrLastEgressId variable of any LTR whose ltrReplyTTL variable contains ***x*** will match the ltrNextEgressId variable of one of the LTRs whose ltrReplyTTL variable contains (***x*** + 1). For LTR entry ***x***, this match indicates which of the Linktrace Responders (***x*** + 1) forwarded the LTM that was received by the Linktrace Responder that returned LTR entry ***x***, and thus turns the list of LTR entries at each ltrReplyTTL value into a tree.
f) Along any given path from the root of the reply tree, the last reply can report any of the following conditions that terminate the forwarding of the LTM:
   1) FwdYes bit of the ltrFlags variable is reset (false); or
   2) TerminalMEP of the ltrFlags variable is set (true); or
   3) ltrRelayAction variable contains the value, RlyHit; or
   4) ltrReplyTTL variable contains 0.
g) If any path does not terminate with one of the conditions 1 through 3 in the previous item f), then either the initial LTM TTL field was insufficiently large to trace the complete path (and the last ltrReplyTTL variable of the end MP in the path contains 0), or a data frame addressed to the target MAC address of the LTM would be flooded after being forwarded from the Bridge that returned the last LTR in the path.
h) If the first Bridge encountered by the original LTM (which is the Bridge on which the LTM was originated, if originated from an Up MEP) would have flooded a data frame addressed to the target MAC address of the LTM, no LTRs will be returned in response to the LTM.
i) A gap in the ltrReplyTTL value sequence indicates the loss of (or failure to transmit) one or more LTRs. Similarly, a gap in the chain of ltrNextEgressId and ltrLastEgressId variables can be caused by the loss of LTRs. These conditions can make the construction of a reply tree impossible, if multiple LTRs have been returned with the same ltrReplyTTL value. If the last LTR along the path is lost, there is no gap in the ltrReplyTTL sequence to indicate the loss.
j) The Transmit Linktrace Message command can be reissued, and the resultant LTRs correlated with the results of a previous LTM, to improve the reliability of the results.

Diagnosing whether an anomalous result from a Read Linktrace Reply command, i.e., one that violates any of the prior assertions, is the result of ordinary frame loss (caused, e.g., by congestion), transitory network behavior (e.g., the loss of a LAN during the Linktrace procedure), a permanent malfunction (e.g., a Bridge is behaving in a noncompliant manner due to a hardware or software malfunction), or some other reason, is beyond the scope of this standard.

J.6 MP addressing: Individual and Shared MP addresses

It is stated in 21.3.2 that the source MAC address for a frame carrying a CFM PDU is the MAC address of the MP transmitting the PDU. This does not necessarily mean that the MP's physical implementation is tied to a particular Bridge Port, and that the MAC addresses used in the frames carrying CFM PDUs are the MAC addresses of those Bridge Ports. While this method of MAC address assignment produces the best protection that this standard can provide for a network, it is not required that every MP use its Bridge Port's MAC address.

CFM does not require the MPs configured on a single Bridge Port, e.g., all of the MPs illustrated in Figure 22-4, to have MAC addresses that are different from each other, nor does CFM require that they be the same. If the MPs in Figure 22-4 all have the same MAC address, CFM frames can be directed to the different MPs on that Bridge Port based on VID and MD Level. Choices can be made, however, with regard to whether MPs on different Bridge Ports can share the same MAC address. Two models are presented in this subclause for the assignment of MAC addresses to MPs:

a) **Individual MP address model**: The Individual MAC address assigned to an MP associated with a particular MA, and hence that MA's MD Level and set of VIDs, is unique over all service instances associated with that same set of VIDs, such that those service instances' CFM PDUs can be distinguished only by their MD Levels that can pass through that MP's Bridge Port.
b) **Shared MP address model**: The same as the Individual MP address model, except that Up MPs (but not Down) in the same MA and in the same Bridge can share a (i.e., can all have the same) MAC address.

Down MPs in different Bridge Ports have different MAC addresses. If two Bridge Ports are connected to the same LAN, but have the same MAC address, then both would respond to an LBM sent to that MAC address.[17] This is true, even though one of those two Bridge Ports would be Blocked (it would be either a Backup or an Alternate Port, see 13.12), because Down MEPs are between the LAN and the Port filtering entities (8.6.1, 8.6.2, 8.6.4) that enforce the blocking. This placement allows Down MEPs to test their service instances even when their Port is blocked, so that the system administrator knows that the failover links will be ready to carry data when a network event causes the Port to become Forwarding.

A Bridge can use the Shared MP address model, or the Individual MP address model, on any given Bridge Port. This is an implementation decision; no managed objects are provided to select one or the other model of operation.

A Bridge using either of these two models can also make use of a third scheme for configuring MPs:

c) **Management Port MEPs**: Up MEPs (not Up MHFs, not Down MPs) are configured on a Management Port, a Bridge Port that does not connect to any LAN exterior to the Bridge. (See 8.3 and Figure 8-7.) Typically, this would be the same MAC address, and the same Bridge Port, that used for the Bridge's Management Port (see Figure 8-7).

The Individual MP address model is described in J.6.1, Shared MP address model in J.6.2, and Management Port MEPs in 22.7.

J.6.1 Individual MP address model

The purpose of CFM is to detect and diagnose connectivity errors. If a Bridge implements its Up MPs as close to the physical layer hardware as possible, it maximizes the capability of CFM to detect connectivity errors caused by malfunctions in that Bridge. In particular, giving each Up MP a MAC address that is tied to its particular Bridge Port means that an LBM/LBR exchange tests the Frame filtering entity (8.6.3) at both ends of the path. Figure 22-5 illustrates this fact. Any LBMs or LBRs passing between the Up MEPs and MHFs at the ends of the grayed service instance traverse exactly the same path through the Frame filtering entity as ordinary data frames carried from the left-most LAN to the right-most LAN.

[17]Bridge implementations are known that use the same source MAC address for different Bridge Ports. As long as the Bridge Port or Management Port to which higher layer entities (e.g., an IP/TCP protocol stack) are attached has its own unique MAC address, and as long as those Bridge Ports that share an address transmit only frames, such as BPDUs that do not traverse other Bridges, and as long as those Ports are not running any protocols that expect unicast frames to pass through a Bridge to reach them, Bridges of this description can interoperate each other and can interoperate with Bridges that follow the requirement to have a different MAC address for every Port. However, CFM PDUs do traverse other Bridges, and the LBM protocol does utilize Individual MAC addresses. Therefore, the CFM protocols do not meet the goals of Clause 18 in all network topologies if Bridge Ports' Down MPs share MAC addresses.

J.6.2 Shared MP address model and the CFM Port

Although the Individual MP address model (J.6.1) gives the best detection and diagnostic capabilities, not all Bridge implementations are capable of inserting and removing CFM PDUs at points close to the physical layer. For example, the reflection of a high-speed LTM/LTR test stream by an Up MEP configured on a Port that is also receiving normal traffic from its LAN implies the presence of prioritization, data path, queuing, and bandwidth resources not otherwise required by this standard. This does not mean that Bridges lacking those resources cannot support CFM. The Shared MP address model provides one way for these Bridges to maximize their CFM capabilities.

Consider Figure 20-13, case c, but assume that the PDU entering Port 5, instead of being an LTM, is an LBM addressed to the Up MEP on Port 3. One cannot discern, from observing the external behavior of a Bridge, whether that LBM was actually delivered to the Up MEP on Port 3 or to some other entity within the Bridge. The Shared MP address model takes advantage of this fact by allowing Up MPs in different Bridge Ports to share the same MAC address. The Up MPs are still configured on different Bridge Ports according to the managed objects in Clause 12. For the most part, they have distinct identities. They share their MAC addresses and, as discussed more fully in 20.48, certain state machines and variables.

In a Bridge configured according to the Shared MP address model, whether an LBM (or other PDU) that enters Port 5 in Figure 20-13 is addressed to the Up MEP on Port 3, or the Up MHF on Port 2, it is delivered to the same physical Bridge Port, called a “CFM Port.” This could be a normal Bridge Port, e.g., one with more capability than the Bridge Port on which the addressed MP is configured, or it could be a Port that is attached to no LAN, in the manner of a Management Port (8.3, 22.7). A Bridge can have more than one CFM Port, but within that Bridge, each CFM Port has a MAC address that is different from the other CFM Ports. In the limiting case, each CFM Port is also a Bridge Port, and is associated with only its own Up MPs; this is indistinguishable from the Individual MP address model. A CFM Port, for which the MPs of three different Bridge Ports are residing, is illustrated in Figure J.1.

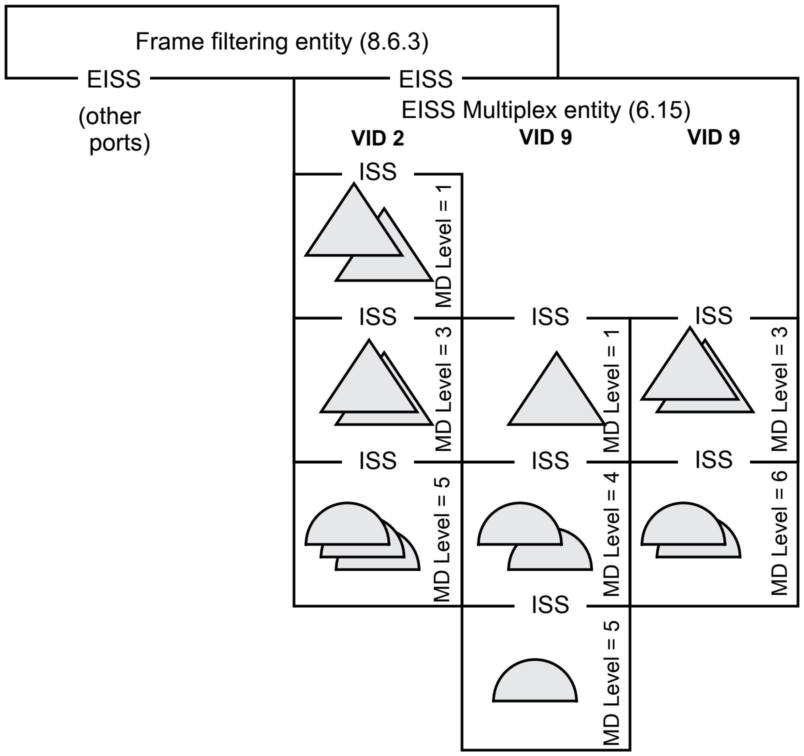

Figure J.1—Up MPs in a CFM Port

NOTE 1—Unlike normal Bridge Ports or a Management Port, Multiple levels of MHFs can be present on a CFM Port for the same service instance, because they belong to different MAs on different Bridge Ports.

MPs on this CFM Port are distinguished by VID and by MD Level. However, those criteria are insufficient when Up MPs configured on different Bridge Ports are in the same MA, and thus in the same VID and at the same MD Level. There is no way of distinguishing, in that case, to which of those Up MPs a given CFM PDU is addressed. For the PDUs defined in this standard, this is not a problem.

Each MEP, even when multiple MEPs reside on a single CFM Port, has its own MEP Continuity Check Initiator state machine (20.12) and associated variables. Each MEP is responsible for transmitting its own CCMs, carrying information (e.g., the optional Sender ID TLV, 21.5.3) peculiar to its configured Bridge Port.

On the other hand, otherwise indistinguishable MEPs sharing a MAC address share a single LTR Transmitter state machine (20.45) and a single MEP Loopback Initiator transmit state machine (20.30), and their associated variables. The managed objects in 12.14.7.1.3 support this sharing.

When any CFM PDU arrives at the CFM Port, it is copied and delivered to each of the MPs that share the same MA. Each of these MPs responds to the CFM PDU separately and responds as if it resided in its configured Bridge Port. Since a multicast would be delivered to all of the MPs anyway, there is no harm in placing all of them in a single CFM Port, with a single MAC address.

Unicast CCMs, LTMs, LBRs, and LTRs present no issues. CCMs require no response. The LTMs all go to the Bridge's single Linktrace Responder. Even when an LTM is multicast, only one LTR and/or forward LTM is produced, so delivering a unicast LTM to multiple MPs causes no additional problems. Since the MEPs sharing a single CFM Port also share a single MEP Loopback Initiator transmit state machine (20.30) and a single MEP Loopback Initiator receive state machine (20.32), and since they all share a single set of variables, particularly those related to the LTM Transaction Identifier fields in these PDUs, there is no confusion as to which MEP the LBR or LTR is directed.

The remaining PDU type is the LBM. Consider what would happen if the MEPs and their associated MP Loopback Responders (19.2.10) resided on separate Bridge Ports, but all shared the same MAC address. Assuming that the Frame filtering entity had learned that MAC address, the LBM would be forwarded to exactly one of those MEPs, namely the one that, by chance, last happened to transmit a frame, thus causing the Learning Process (8.7) to associate the shared MAC address with that transmitter's Bridge Port. Thus, exactly one LBR would be returned, not one per Bridge Port. The MEP receiving that LBR would not know from which MP the LBR was returned, because there is nothing in the LBR to indicate the identity of the responding MP. Therefore, only one response to a unicast LBM is returned from a CFM Port, no matter how many MPs on that LBM's MA reside on the CFM Port.

On the other hand, a multicast LBM could cause any number of Up MPs configured in the same MA, but on different Bridge Ports in a single Bridge, to respond. Thus, a high-speed stream of multicast LBMs could impose an arbitrarily large burden upon a single CFM Port that is supporting a large number of Bridge Ports' MPs, if each MP responded to that LBM. Therefore, among a set of MPs for a single MA that all share the same MAC address (i.e., reside on the same CFM Port), one or more, but not all, of their MP Loopback Responders (19.2.10) can be disabled by not configuring any Group MAC address for them to recognize. This configuration is accomplished automatically by a Bridge that utilizes CFM Ports and is not controlled by any managed object.

The externally visible behaviors that can be discerned from outside a Bridge that uses the Shared MP address model are:

a) An external MEP can regularly receive CFM PDUs other than LTRs, from two different MPs in the same MA that have the same source_address.

b) A multicast LBM can regularly receive only one reply from among a set of MPs that normally all respond to unicast LBMs, but all with the same source_address, without implying the presence of an intermittent network failure.
c) A system administrator can see unexpected advances in a MEP's managed objects that count LBMs and LBRs, apparently due to activity on another MEP.
d) If conceptual rows in the Interface MIB module of IETF RFC 2863 are instantiated for individual MPs, the packet and frame counts of those interfaces count both CFM PDUs and ordinary data frames passing through the Active and Passive SAPs, and thus can indicate whether an MP is instantiated on the Bridge Port on which it is configured, or on a separate CFM Port through which no ordinary data frames pass.

NOTE 2—None of these behaviors is impossible for a Bridge using the Individual MP address model; MAC addresses can change faster than CCMs time out, momentary congestion can cause frame loss, multiple administrators can access different MEPs' managed objects, and ordinary data frames can be absent. It is only the frequency of these behaviors that distinguishes the Individual MP address model from the Shared MP address model.