

**IEEE Standard for
Local and metropolitan area networks—**

**Media Access Control (MAC) Bridges and
Virtual Bridge Local Area Networks**

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

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

31 August 2011

IEEE Std 802.1Q™-2011
(Revision of
IEEE Std 802.1Q-2005)

**IEEE Standard for
Local and metropolitan area networks—**

**Media Access Control (MAC) Bridges and
Virtual Bridge Local Area Networks**

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

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

31 August 2011

IEEE Std 802.1Q™-2011
(Revision of
IEEE Std 802.1Q-2005)

IEEE Std 802.1Q™-2011
(Revision of
IEEE Std 802.1Q-2005)

**IEEE Standard for
Local and metropolitan area networks—**

Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks

Sponsor

**LAN/MAN Standards Committee
of the
IEEE Computer Society**

Approved 16 May 2011

IEEE-SA Standards Board

Abstract: This standard specifies how the MAC Service is supported by Virtual Bridged Local Area Networks, the principles of operation of those networks, and the operation of VLAN-aware Bridges, including management, protocols, and algorithms.

Keywords: Bridged Local Area Networks, local area networks (LANs), MAC Bridges, metropolitan area networks, Multiple Spanning Tree Protocol (MSTP), Rapid Spanning Tree Protocol (RSTP), Virtual Bridged Local Area Networks (virtual LANs)

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA
Copyright © 2011 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 31 August 2011. Printed in the United States of America.

IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-0-7381-6708-4 STD97139
Print: ISBN 978-0-7381-6709-1 STDPD97139

*IEEE prohibits discrimination, harassment and bullying. For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.
No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.*

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied **“AS IS.”**

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation, or every ten years for stabilization. When a document is more than five years old and has not been reaffirmed, or more than ten years old and has not been stabilized, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Recommendations to change the status of a stabilized standard should include a rationale as to why a revision or withdrawal is required. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854-4141
USA

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

This introduction is not part of IEEE Std 802.1Q-2011, IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks.

The MAC Bridge standardization activities that resulted in the development of IEEE Std 802.1DTM, 1993 Edition [B9],^a introduced the concept of Filtering Services in Bridged Local Area Networks, and mechanisms whereby filtering information in such LANs may be acquired and held in a Filtering Database.

IEEE Std 802.1D, 1998 Edition, a revision of IEEE Std 802.1D, 1993 Edition, extended this concept of Filtering Services to define additional capabilities aimed at

- a) The provision of expedited traffic capabilities, to support the transmission of time-critical information in a LAN environment.
- b) The use of signaled priority information as the basis for identifying expedited classes of traffic.
- c) The provision of filtering services that support the dynamic definition and establishment of Groups in a LAN environment, and the filtering of frames by Bridges such that frames addressed to a particular Group are forwarded only on those LANs that are required to reach members of that Group.
- d) The provision of a Generic Attribute Registration Protocol (GARP) that is used to support the mechanism for providing Group filtering capability and is made available for use in other attribute registration applications.

This standard, first published as IEEE Std 802.1Q-1998, makes use of the concepts and mechanisms of LAN Bridging that were introduced by IEEE Std 802.1D, and it defines additional mechanisms that allow the implementation of Virtual Bridged Local Area Networks. The following mechanisms are described:

- e) Virtual LAN Services.
- f) The operation of the Forwarding Process that is required.
- g) The structure of the Filtering Database that is required.
- h) The nature of the protocols and procedures that are required to provide Virtual LAN services, including the definition of the frame formats used to represent VLAN identification information, and the procedures used to insert and remove VLAN identifiers (VIDs) and the headers in which they are carried.
- i) 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.
- j) The management services and operations that are required to configure and administer networks.

The 2005 Revision of the standard incorporated three amendments, IEEE Std 802.1uTM-2001, IEEE Std 802.1vTM-2001, and IEEE Std 802.1sTM-2002, into the text of IEEE Std 802.1Q-1998. These amendments describe enhancements to the standard to allow

- k) Dynamic Group and VLAN registration to be restricted, based on the contents of static filtering entries.
- l) VLAN classification according to link layer protocol type.
- m) Support for VLANs carried over multiple Spanning Tree instances.

^aNumbers in brackets correspond to the number in the bibliography in Annex M.

This revision of the standard incorporates amendments IEEE Std 802.1adTM-2005, IEEE Std 802.1akTM-2007, IEEE Std 802.1agTM-2007, IEEE Std 802.1ahTM-2008, IEEE Std 802-1Q-2005/Cor-1-2008, IEEE Std 802.1apTM-2008, IEEE Std 802.1QawTM-2009, IEEE Std 802.1QayTM-2009, IEEE Std 802.1ajTM-2009, IEEE Std 802.1QavTM-2009, IEEE Std 802.1QauTM-2010, and IEEE Std 802.1QatTM-2010 into the text of IEEE Std 802.1Q-2005. These amendments describe enhancements to the standard to allow

- n) A service provider to use the architecture and protocols defined in this standard to offer the equivalent of separate LANs, Bridged Local Area Networks, or Virtual Bridged Local Area Networks to a number of customers, while requiring no cooperation between the customers, and minimal cooperation between each customer and the service provider.
- o) The distribution and registration of VLAN membership information using the Multiple VLAN Registration Protocol (MVRP), and the distribution of group membership information using the Multiple MAC Registration Protocol (MMRP), replacing the previous protocols, GVRP and GMRP.
- p) Support of Connectivity Fault Management (CFM) capabilities for detecting, isolating, and reporting connectivity faults.
- q) Interconnection of multiple Provider Bridged Networks, allowing a Provider to support up to 2^{24} service instances.
- r) Management of VLAN Bridges via SNMP MIB modules.
- s) Diagnosis of data-dependent connectivity faults.
- t) Support for traffic engineering.
- u) The use of Bridging technology as a two-port MAC relay.
- v) Support for the relay of time sensitive data streams.
- w) Support for congestion management of long-lived data flows within network domains of limited bandwidth-delay product. This is achieved by enabling bridges to signal congestion to end stations capable of transmission rate limiting to avoid frame loss.
- x) Network resources to be reserved for specific traffic streams traversing a bridged local area network.
- y) The use of IEEE 802.17TM and IEEE 802.20TM media in bridged local area networks.

As part of this revision of the standard, Clause 13 was extensively revised in order to integrate the description of the Rapid Spanning Tree Protocol (RSTP) into the text of the standard rather than including it by reference to IEEE Std 802.1D.

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 may be obtained from

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854-4141
USA

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association website at

<http://ieeexplore.ieee.org/xpl/standards.jsp>, or contact the IEEE at the address listed previously. For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA website at <http://standards.ieee.org>.

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. The IEEE shall not be responsible for identifying patents or patent applications for which a license may be required to implement an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention. 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 and nondiscriminatory, reasonable terms and conditions to applicants desiring to obtain such licenses. The IEEE makes no representation as to the reasonableness of rates, terms, and conditions of the license agreements offered by patent holders or patent applicants. Further information may be obtained from the IEEE Standards Department.

Participants

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

Anthony Jeffree, Chair and Editor
Paul Congdon, Vice Chair
Stephen Haddock, Chair, Interworking Task Group

Zehavit Alon	Anoop Ghanwani	Donald Pannell
Yafan An	Eric Gray	Glenn Parsons
Ting Ao	Yingjie Gu	Joseph Pelissier
Peter Ashwood-Smith	Craig Gunther	Rene Raeber
Christian Boiger	Hitoshi Hayakawa	Karen Randall
Paul Bottorff	Hal Keen	Dan Romascanu
Rudolf Brandner	Yongbum Kim	Jessy Rouyer
Craig Carlson	Philippe Klein	Panagiotis Saltsidis
Claudio Desanti	Oliver Kleineberg	Michael Seaman
Zhemin Ding	Michael Krause	Rakesh Sharma
Donald Eastlake	Li Li	Kevin B. Stanton
Janos Farkas	Jeff Lynch	Robert Sultan
Donald Fedyk	Thomas Mack-Crane	Michael Teener
Norman Finn	John Messenger	Patricia Thaler
Ilango Ganga	John Morris	Chait Tumuluri
Geoffrey Garner	Eric Multanen	Maarten Vissers
	David Olsen	

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

Thomas Alexander	C. Huntley	Maximilian Riegel
Zehavit Alon	Akio Iso	Robert Robinson
Danilo Antonelli	Atsushi Ito	Jessy Rouyer
Hugh Barrass	Raj Jain	Randall Saffier
Robert Boatright	Junghoon Jee	Panagiotis Saltsidis
Tomo Bogataj	Anthony Jeffree	Peter Saunderson
Gennaro Boggia	Shinkyo Kaku	Bartien Sayogo
Nancy Bravin	Piotr Karocki	Benson Schlieser
William Byrd	Stuart J. Kerry	Michael Seaman
Juan Carreon	Yongbum Kim	Rich Seifert
Keith Chow	Bruce Kraemer	Gil Shultz
Charles Cook	John Lemon	Kapil Sood
Wael Diab	Michael Lerer	Matthew Squire
Russell Dietz	Li Li	Kevin B. Stanton
Thomas Dineen	William Lumpkins	Thomas Starai
Sourav Dutta	Greg Luri	Walter Struppler
Jose Dominic Espejo	Thomas Mack-Crane	Robert Sultan
Donald Fedyk	Elvis Maculuba	Joseph Tardo
Prince Francis	Arthur Marris	William Taylor
Yukihiro Fujimoto	Jeffery Masters	Patricia Thaler
Ilango Ganga	Jonathon McLendon	David Thompson
Randall Groves	Joseph Moran	Geoffrey Thompson
Craig Gunther	Michael S. Newman	Jeffrey Towne
C. Guy	Nick S. A. Nikjoo	Prabodh Varshney
Stephen Haddock	Satoshi Obara	Ludwig Winkel
Marek Hajduczenia	David Olsen	Michael Wright
David Hunter	Glenn Parsons	Oren Yuen

When the IEEE-SA Standards Board approved this standard on 16 May 2011, it had the following membership:

Richard H. Hulett, Chair
John Kulick, Vice Chair
Robert M. Grow, Past President
Judith Gorman, Secretary

Masayuki Ariyoshi
William Bartley
Ted Burse
Clint Chaplin
Weal Diab
Jean-Philippe Faure
Alexander Gelman
Paul Houzé

Jim Hughes
Joseph L. Koepfinger*
David J. Law
Thomas Lee
Hung Ling
Oleg Logvinov
Ted Olsen

Gary Robinson
Jon Walter Rosdahl
Sam Sciacca
Mike Seavey
Curtis Siller
Phil Winston
Howard Wolfman
Don Wright

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Michelle Turner
IEEE Standards Program Manager, Document Development

Kathryn Bennett
IEEE Standards Program Manager, Technical Program Development

Historical participants

Since the initial publication, many IEEE standards have added functionality or provided updates to material included in this standard. The following is a historical list of participants who have dedicated their valuable time, energy, and knowledge to the creation of this material:

IEEE 802.1Q Standard	Date approved by IEEE	Officers at the time of Working Group Letter Ballot
IEEE Std 802.1Q-1998	8 December 1998	William P. Lidinsky , <i>Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i> Tony Jeffree , <i>Coordinating Editor</i> Anil Rijsinghani, Richard Hausmann, Michele Wright, Paul Langille, P. J. Singh , <i>Editorial Team</i>
IEEE Std 802.1u-2001	17 March 2001	Tony Jeffree , <i>Chair</i> Neil Jarvis , <i>Vice Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i>
IEEE Std 802.1v-2001	17 March 2001	Tony Jeffree , <i>Chair</i> Neil Jarvis , <i>Vice Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i> David Delany , <i>Editor</i> Andrew Smith , <i>Editor</i>
IEEE Std 802.1s-2002	11 December 2002	Tony Jeffree , <i>Chair</i> Neil Jarvis , <i>Vice Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i> Norman W. Finn , <i>Editor</i>
IEEE Std 802.1ad-2005	28 March 2005	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i> Stephen R. Haddock , <i>Editor</i>
IEEE Std 802.1Q-2005	7 December 2005	Tony Jeffree , <i>Chair and Editor</i> Paul Congdon , <i>Vice Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i>
IEEE Std 802.1ak-2007	22 March 2007	Tony Jeffree , <i>Chair and Editor</i> Paul Congdon , <i>Vice Chair</i> Mick Seaman , <i>Chair; Interworking Task Group</i>

IEEE 802.1Q Standard	Date approved by IEEE	Officers at the time of Working Group Letter Ballot
IEEE Std 802.1ag-2007	27 September 2007	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i> Norman W. Finn , <i>Editor-in-Chief</i> David V. Elie-Dit-Cosaque , Dinesh Mohan , Oscar Rodriguez , and Ali Sajassi , <i>Assistant Editors</i>
IEEE Std 802.1ah-2008	12 June 2008	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i> Paul Bottorff , Stephen Haddock , and Muneyoshi Suzuki , <i>Editors</i>
IEEE Std 802.1Q-2005/ Cor-1-2008	26 September 2008	Tony Jeffree , <i>Chair and Editor</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i>
IEEE Std 802.1ap-2008	10 December 2008	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i> Glenn Parsons , <i>Editor</i> David Levi , <i>Assistant Editor</i>
IEEE Std 802.1Qaw-2009	17 June 2009	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i> Linda Dunbar , <i>Editor</i>
IEEE Std 802.1Qay:2009	17 June 2009	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i> Panagiotis Saltsidis , <i>Editor</i>
IEEE Std 802.1aj:2009	9 December 2009	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Stephen R. Haddock , <i>Chair; Interworking Task Group</i> John Messenger , <i>Editor</i> Brian Hassink , <i>MIB Editor</i>

IEEE 802.1Q Standard	Date approved by IEEE	Officers at the time of Working Group Letter Ballot
IEEE Std 802.1Qav-2009	9 November 2009	Tony Jeffree , <i>Chair and Editor</i> Paul Congdon , <i>Vice Chair</i> Michael Johas Teener , <i>Chair; Audio Video Bridging Task Group</i>
IEEE Std 802.1Qau-2010	25 March 2010	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Patricia Thaler , <i>Chair; Data Center Bridging Task Group</i> Norman W. Finn , <i>Editor</i>
IEEE Std 802.1Qat-2010	30 September 2010	Tony Jeffree , <i>Chair</i> Paul Congdon , <i>Vice Chair</i> Michael Johas Teener , <i>Chair; Audio Video Bridging Task Group</i> Craig Gunther , <i>Editor</i>

Osama Aboul-Magd	Prakash Desai	David Husak
Steve Adams	Claudio Desanti	Altaf Hussain
Stephen Ades	Jeffrey Dietz	Romain Insler
Zehavit Alon	Russell Dietz	Ran Ish-Shalom
Ken Alonge	Zhemin Ding	Vipin K. Jain
Ting Ao	Kurt Dobbins	Tony Jeffree
Siamack Ayandeh	Linda Dunbar	Paul Hongkyu Jeong
Floyd Backes	Craig Easley	Michael Johas Teener
John Bartlett	Donald Eastlake, III	Pankaj Jha
Alexei Beliaeav	Peter Ecclesine	Shyam Kaluve
Les Bell	J. J. Ekstrom	Abhay Karandikar
Avner Ben-Dor	Anush Elangovan	Allen Kasey
Michael Berger	Hesham M. Elbakoury	Prakash Kashyap
Caitlin Bestler	David Elie-Dit-Cosaque	Toyayuki Kato
Jan Bialkowski	Janos Farkas	Hal Keen
James S. Binder	Donald W. Fedyk	Daniel Kelley
Rob Boatright	Felix Feifei Feng	Kevin Ketchum
Jean-Michel Bonnamy	Norman Finn	Keti Kilcrease
Mike Borza	Yishai Fraenkel	Doyeon Kim
Paul Bottorff	Paul Frantz	Yongbum Kim
David Brady	David Frattura	Keith Klamm
Rudolf Brandner	Robert Frazier	Philippe Klein
Martin Brewer	Lars Henrik Frederiksen	Bruce Kling
Bill Bunch	John Fuller	Walter Knitl
Jim Burns	Geoffrey Garner	Mike Ko
Bob Cardinal	Anoop Ghanwani	Raghav Kondapalli
Craig W. Carlson	Franz Goetz	Michael Krause
Paul Carroll	Eric Gray	Dan Krent
Jeffrey Catlin	Karanvir Grewal	Vinod Kumar
Dennis Cave	John Grinham	Paul Kummer
Dirceu Cavendish	Craig Gunther	Bruce Kwan
Alan Chambers	Mitch Gusat	Paul Lachapelle
Steve Chan	Stephen Haddock	Kari Laihonen
David W. Chang	Sharam Hakimi	Ashvin Lakshmikantha
Frank Chao	John Hart	Bill Lane
Ken Chapman	Scott Harvell	Loren Larsen
Weiying Cheng	Takashi Hasegawa	Yannick Le Goff
Rao Cherukuri	Brian Hassink	John Lemon
Taesik Cheung	Wayne Hathaway	Michael Lerer
Hon Wah Chin	Brian Hausauer	Johann Lindmeyr
Jin-Seek Choi	Vic Hayes	Marina Lipshteyn
Chi Chong	Asif Hazarika	Gary Littleton
Chris Christ	David Head	Yuanqui Luo
Marc Cochran	Gaby Hecht	Robert D. Love
Paul Congdon	Deepak Hegde	Andy Luque
Glenn Connery	Ariel Hendel	Gael Mace
Alex Conta	John Hickey	Ben Mack-Crane
Diego Crupnicoff	David Hollender	David Martin
David Cullerot	Steve Horowitz	Peter Martini
Uri Cummings	Bob Hott	Riccardo Martinotti
Ted Davies	Michelle Hsiung	Marco Mascitto
Andy Davis	Charles Hudson	Keith McCloghrie
Arjan de Heer	Rita Hunt	Alan McGuire

James McIntosh	Dick Reohr	George Swallow
Martin McNealis	James Richmond	Richard Sweatatt
Menucher Menuchery	Robert Roden	Attila Takacs
Milan Merhar	Guenter Roeck	Francois Tallet
John Messenger	John J. Roese	Robin Tasker
Colin Mick	Josef Roese	John Terry
Amol Mitra	Derek J. Rohde	Patricia A. Thaler
Dinesh Mohan	Allyn Romanow	Geoff Thompson
Gabriel Montenegro	Dan Romascanu	Oliver Thorp
Matthew Mora	Moran Roth	Michel Thorsen
John Morris	Jessy V. Rouyer	Fouad Tobagi
Bob Moskowitz	Doug Ruby	Naoki Tsukutari
Eric Multanen	Eric Ryu	Fred Tuck
Yaron Nachman	Jonathan Sadler	Paul Unbehagen
Krishna Narayanaswamy	Ali Sajassi	Dhadesugor Vaman
Lawrence Ng	Joseph Salowey	Steve Van Seters
Henry Ngai	Panagiotis Saltsidis	Dono van-Mierop
Paul Nikolich	Sam Sambasivan	John Viega
Kevin Nolish	Ray Samora	Maarten Vissers
Don O'Connor	Satish Sathe	Dennis Volpano
Karen O'Donoghue	John M. Sauer	Manoj Wadekar
Eugene O'Neil	Ayman Sayed	Yuehua Wei
Satoshi Obara	Ted Schroeder	John Wakerly
Hiroshi Ohta	Benjamin Schultz	Peter Wang
David Olsen	Michael J. Seaman	Philip Wang
Toshio Ooka	Rich Seifert	Y. C. Wang
Jörg Ottensmeyer	Lee Sendelbach	Yan Wang
Shlomo Ovadia	Koichiro Seto	Trevor Warwick
Vijoy Pandey	Himanshu Shah	Bob Watson
Don Pannell	Rakesh Sharma	Karl Weber
Luc Pariseau	Ravi Shenoy	Brian Weis
Glenn W. Parsons	K. Karl Shimada	Alan Weissberger
Ken Patton	Fred Shu	Glenn Wenig
Joe Pelissier	Taeshi Shimizu	Martin White
Yonadav Perry	Phil Simmons	Bert Wijnen
David Peterson	Curtis Simonson	Keith Willette
John Pickens	Rosemary V. Slager	Robert Williams
Hayim Porat	Alexander Smith	Ludwig Winkel
Gideon Prat	Michel Soerensen	Robert Winter
Kirk Preiss	Stuart Soloway	Michael Witkowski
Max Pritikin	Nurit Sprecher	Edward Wong
Ray Qiu	Kevin B. Stanton	Michael D. Wright
Ananda Rajagopal	Larry Stefani	Chien-Hsien Wu
Steve Ramberg	Sundar Subramaniam	Ken Young
Karen Randall	Robert Sultan	Allen Yu
Shlomo Reches	Muneyoshi Suzuki	Wayne Zakowski
Frank Reichstein	Yoshihiro Suzuki	Glen Zorn

Contents

1.	Overview.....	1
1.1	Scope.....	2
1.2	Purpose.....	2
1.3	Introduction.....	2
1.4	VLAN aims and benefits	7
2.	Normative references.....	8
3.	Definitions	11
4.	Abbreviations.....	26
5.	Conformance.....	30
5.1	Requirements terminology	30
5.2	Conformant components and equipment	30
5.3	Protocol Implementation Conformance Statement (PICS).....	31
5.4	VLAN-aware Bridge component requirements	31
5.4.1	VLAN-aware Bridge component options	32
5.4.2	Multiple VLAN Registration Protocol (MVRP) requirements	35
5.4.4	Multiple Stream Registration Protocol (MSRP) requirements	36
5.5	C-VLAN component conformance.....	37
5.5.1	C-VLAN component options	37
5.6	S-VLAN component conformance	37
5.6.1	S-VLAN component options	38
5.6.2	S-VLAN component requirements for PBB-TE	38
5.7	I-component conformance	38
5.7.1	I-component options	38
5.8	B-component conformance.....	39
5.8.1	B-component options	39
5.8.2	B-component requirements for PBB-TE	39
5.9	VLAN Bridge conformance.....	40
5.9.1	VLAN Bridge options	40
5.10	Provider Bridge conformance	40
5.10.1	S-VLAN Bridge conformance	40
5.10.2	Provider Edge Bridge conformance	40
5.11	Backbone Edge Bridge conformance	41
5.11.1	Backbone Edge Bridge requirements for PBB-TE	41
5.12	VLAN-unaware Bridge component requirements.....	41
5.13	TPMR component conformance.....	41
5.13.1	TPMR component options	42
5.14	TPMR conformance.....	42
5.14.1	TPMR options	42
5.15	T-component conformance	42
5.15.1	T-component options	43
5.16	End station requirements for MMRP, MVRP, and MSRP	43
5.16.1	MMRP requirements and options	43
5.16.2	MVRP requirements and options	43
5.16.3	MSRP requirements and options	44

5.17	VLAN-aware end station requirements for Connectivity Fault Management.....	44
5.18	End station requirements—forwarding and queuing for time-sensitive streams.....	45
5.19	End station requirements for congestion notification	45
5.20	MAC-specific bridging methods	46
6.	Support of the MAC Service	47
6.1	Basic architectural concepts and terms	48
6.1.1	Protocol entities, peers, layers, services, and clients	48
6.1.2	Service interface primitives, parameters, and frames	48
6.1.3	Layer management interfaces	49
6.1.4	Service access points, interface stacks, and ports	49
6.1.5	MAC method independent protocols and shims	50
6.1.6	MAC Service clients	50
6.1.7	Stations and systems	51
6.1.8	Connectionless connectivity	51
6.2	Provision of the MAC service	51
6.2.1	Point-to-point, multipoint-to-multipoint, and rooted-multipoint connectivity	52
6.3	Support of the MAC service	52
6.4	Preservation of the MAC service.....	53
6.5	Quality of service maintenance.....	53
6.5.1	Service availability	53
6.5.2	Frame loss	54
6.5.3	Frame misordering	54
6.5.4	Frame duplication	55
6.5.5	Transit delay	55
6.5.6	Frame lifetime	56
6.5.7	Undetected frame error rate	56
6.5.8	Maximum Service Data Unit Size	56
6.5.9	Priority	56
6.5.10	Throughput	57
6.6	Internal Sublayer Service.....	57
6.6.1	Service primitives and parameters	58
6.6.2	Status parameters	60
6.6.3	Point-to-point parameters	60
6.6.4	Stream Reservation Protocol (SRP) Domain status parameters	60
6.7	Support of the Internal Sublayer Service by specific MAC procedures.....	61
6.7.1	Support of the Internal Sublayer Service by IEEE Std 802.3 (CSMA/CD)	61
6.7.2	Support by IEEE Std 802.11 (Wireless LAN)	63
6.7.3	Support by IEEE Std 802.17 (RPR)	64
6.7.4	Support by IEEE 802.20 Mobile Broadband Wireless Access Method (MBWA)	65
6.8	Enhanced Internal Sublayer Service.....	67
6.8.1	Service primitives	67
6.8.2	Status parameters	68
6.8.3	Point-to-point parameters	68
6.9	Support of the EISS	68
6.9.1	Data indications	69
6.9.2	Data requests	70
6.9.3	Priority Code Point encoding	70
6.9.4	Regenerating priority	72
6.10	Support of the ISS/EISS by Provider Instance Ports	73
6.10.1	Data indications	75
6.10.2	Data requests	76
6.10.3	Priority Code Point encoding	77

6.11	Support of the EISS by Customer Backbone Ports.....	77
6.11.1	Data indications	78
6.11.2	Data requests	79
6.11.3	Priority Code Point decoding	80
6.11.4	Regenerating priority	80
6.12	Protocol VLAN classification.....	80
6.12.1	Protocol Templates	82
6.12.2	Protocol Group Identifiers	82
6.12.3	Protocol Group Database	83
6.13	Support of the ISS for attachment to a Provider Bridged Network.....	83
6.13.1	Data requests	84
6.13.2	Data indications	85
6.14	Support of the ISS within a system.....	85
6.15	Support of the ISS by additional technologies.....	86
6.16	Filtering services in Bridged Local Area Networks	86
6.16.1	Purpose(s) of filtering service provision	86
6.16.2	Goals of filtering service provision	87
6.16.3	Users of filtering services	87
6.16.4	Basis of service	87
6.16.5	Categories of service	87
6.16.6	Service configuration	88
6.16.7	Service definition for Extended Filtering Services	88
6.17	EISS Multiplex Entity.....	90
6.18	Backbone Service Instance Multiplex Entity.....	91
6.18.1	Demultiplexing direction	92
6.18.2	Multiplexing direction	93
6.18.3	Priority Code Point encoding	93
6.18.4	Status parameters	94
6.19	TESI Multiplex Entity	94
6.20	Support of the ISS with signaled priority	95
6.20.1	Data indications	96
6.20.2	Data requests	96
7.	Principles of network operation	97
7.1	Network overview.....	97
7.2	Use of VLANs	98
7.3	VLAN topology	99
7.4	Locating end stations	99
7.5	Ingress, forwarding, and egress rules.....	101
8.	Principles of bridge operation.....	103
8.1	Bridge operation	103
8.1.1	Relay	103
8.1.2	Filtering and relaying information	104
8.1.3	Duplicate frame prevention	104
8.1.4	Traffic segregation	104
8.1.5	Traffic reduction	105
8.1.6	Traffic expediting	105
8.1.7	Conversion of frame formats	105
8.2	Bridge architecture.....	106
8.3	Model of operation.....	107
8.4	Active topologies, learning, and forwarding	110
8.5	Bridge Port Transmit and Receive.....	111

8.5.1	Bridge Port connectivity	111
8.5.2	TPMR Port connectivity	112
8.5.3	Support of Higher Layer Entities	113
8.6	The Forwarding Process	113
8.6.1	Active topology enforcement	114
8.6.2	Ingress filtering	115
8.6.3	Frame filtering	115
8.6.4	Egress	117
8.6.5	Flow classification and metering	117
8.6.6	Queuing frames	118
8.6.7	Queue management	119
8.6.8	Transmission selection	120
8.7	The Learning Process.....	122
8.7.1	Default filtering utility criteria	122
8.7.2	Enhanced filtering utility criteria	122
8.8	The Filtering Database.....	122
8.8.1	Static Filtering Entries	125
8.8.2	Static VLAN Registration Entries	126
8.8.3	Dynamic Filtering Entries	127
8.8.4	MAC Address Registration Entries	128
8.8.5	Dynamic VLAN Registration Entries	128
8.8.6	Default Group filtering behavior	128
8.8.7	Dynamic Reservation Entries	130
8.8.8	Allocation of VIDs to FIDs	130
8.8.9	Querying the Filtering Database	133
8.8.10	Determination of the member set for a VID	136
8.8.11	Permanent Database	136
8.8.12	Connection_Identifier	137
8.9	MST and ESP configuration information	137
8.9.1	MST Configuration Table	138
8.9.2	MST configuration identification	138
8.9.3	FID to MSTI Allocation Table	138
8.10	Spanning Tree Protocol Entity.....	138
8.11	MRP Entities.....	139
8.12	Bridge Management Entity.....	139
8.13	Addressing	139
8.13.1	End stations	139
8.13.2	Bridge Ports	139
8.13.3	Use of LLC by Spanning Tree Protocol Entities	139
8.13.4	Reserved MAC Addresses	140
8.13.5	Group MAC Addresses for spanning tree protocols	140
8.13.6	Group MAC Addresses for MRP Applications	141
8.13.7	Bridge Management Entities	141
8.13.8	Unique identification of a Bridge	141
8.13.9	Points of attachment and connectivity for Higher Layer Entities	142
8.13.10	VLAN attachment and connectivity for Higher Layer Entities	145
8.13.11	Connectivity Fault Management entities	146
9.	Tagged frame format	148
9.1	Purpose of tagging	148
9.2	Representation and encoding of tag fields.....	148
9.3	Tag format.....	149
9.4	Tag Protocol Identifier (TPID) formats	149

9.5	Tag Protocol Identification	149
9.6	VLAN Tag Control Information.....	150
9.7	Backbone Service Instance Tag Control Information.....	151
10.	Multiple Registration Protocol (MRP) and Multiple MAC Registration Protocol (MMRP)	153
10.1	MRP overview	153
10.2	MRP architecture	156
10.3	MRP Attribute Propagation (MAP).....	157
10.3.1	MAP Context	158
10.4	Requirements to be met by MRP	159
10.5	Requirements for interoperability between MRP Participants	159
10.6	Protocol operation.....	161
10.7	Protocol specification	165
10.7.1	Notational conventions and abbreviations	165
10.7.2	Registrar Administrative Controls	167
10.7.3	Applicant Administrative Controls	168
10.7.4	Protocol timers	168
10.7.5	Protocol event definitions	168
10.7.6	Protocol Action definitions	171
10.7.7	Applicant state machine	173
10.7.8	Registrar state machine	175
10.7.9	LeaveAll state machine	175
10.7.10	PeriodicTransmission state machine	175
10.7.11	Timer values	176
10.7.12	Operational reporting and statistics	177
10.7.13	Interoperability considerations	177
10.8	Structure and encoding of MRP Protocol Data Units.....	178
10.8.1	Structure	178
10.8.2	Encoding of MRPDU parameters	180
10.8.3	Packing and parsing MRPDUs	183
10.9	Multiple MAC Registration Protocol (MMRP)—Purpose.....	184
10.10	Model of operation.....	185
10.10.1	Propagation of Group Membership information	186
10.10.2	Propagation of Group service requirement information	187
10.10.3	Source pruning	188
10.10.4	Use of Group service requirement registration by end stations	188
10.11	Default Group filtering behavior and MMRP propagation	188
10.12	Definition of the MMRP application.....	190
10.12.1	Definition of MRP elements	190
10.12.2	Provision and support of Extended Filtering Services	192
10.12.3	Use of “new” declaration capability	194
10.12.4	Attribute value support requirements	194
11.	VLAN topology management.....	195
11.1	Static and dynamic VLAN configuration	195
11.2	Multiple VLAN Registration Protocol	196
11.2.1	MVRP overview	196
11.2.2	VLAN registration service definition	198
11.2.3	Definition of the MVRP application	199
11.2.4	VID translation table	201
11.2.5	Use of “new” declaration capability	201
11.2.6	Attribute value support requirements	202

12. Bridge management	203
12.1 Management functions.....	203
12.1.1 Configuration Management	203
12.1.2 Fault Management	204
12.1.3 Performance Management	204
12.1.4 Security Management	204
12.1.5 Accounting Management	204
12.2 VLAN-aware bridge objects	204
12.3 Data types	205
12.4 Bridge Management Entity	206
12.4.1 Bridge Configuration	206
12.4.2 Port configuration	208
12.5 MAC entities.....	209
12.6 Forwarding process.....	209
12.6.1 The Port Counters	209
12.6.2 Priority handling	210
12.6.3 Traffic Class Table	218
12.7 Filtering Database	218
12.7.1 The Filtering Database	219
12.7.2 A Static Filtering Entry	220
12.7.3 A Dynamic Filtering Entry	220
12.7.4 A MAC Address Registration Entry	220
12.7.5 A VLAN Registration Entry	220
12.7.6 Permanent Database	221
12.7.7 General Filtering Database operations	221
12.8 Bridge Protocol Entity	223
12.8.1 The Protocol Entity	223
12.8.2 Bridge Port	227
12.9 MRP Entities.....	230
12.9.1 The MRP Timer object	231
12.9.2 The MRP Attribute Type object	231
12.9.3 Periodic state machine objects	232
12.10 Bridge VLAN managed objects.....	233
12.10.1 Bridge VLAN Configuration managed object	233
12.10.2 VLAN Configuration managed object	237
12.10.3 The VLAN Learning Constraints managed object	238
12.11 MMRP entities	243
12.11.1 MMRP Configuration managed object	243
12.12 MST configuration entities	244
12.12.1 The MSTI List	244
12.12.2 The FID to MSTID Allocation Table	246
12.12.3 The MST Configuration Table	247
12.13 Provider Bridge management	248
12.13.1 Provider Bridge Port Type managed object	250
12.13.2 Network Port Configuration managed object	250
12.13.3 Customer Edge Port Configuration managed object	251
12.14 CFM entities	255
12.15 Backbone Core Bridge management	271
12.16 Backbone Edge Bridge management.....	271
12.16.1 BEB configuration managed object	273
12.16.2 BEB/PB/VLAN Bridge Port configuration managed object	276
12.16.3 VIP configuration managed object	276
12.16.4 PIP configuration managed object	278

12.16.5	CBP Configuration managed object	284
12.17	DDCFM entities.....	286
12.17.1	DDCFM Stack managed object	286
12.17.2	Reflection Responder managed object	287
12.17.3	RFM Receiver managed object	290
12.17.4	Decapsulator Responder managed object	291
12.17.5	SFM Originator managed object	294
12.18	PBB-TE Protection Switching managed objects	296
12.18.1	TE protection group list managed object	296
12.19	TPMR managed objects.....	300
12.19.1	TPMR management entity	300
12.19.2	MAC and PHY entities	302
12.19.3	Forwarding Process	302
12.19.4	MAC status propagation entity	308
12.20	Management entities for forwarding and queueing for time-sensitive streams.....	310
12.20.1	The Bandwidth Availability Parameter Table	310
12.20.2	The Transmission Selection Algorithm Table	310
12.20.3	The Priority Regeneration Override Table	311
12.21	Congestion notification managed objects	311
12.22	SRP entities.....	315
12.22.1	SRP Bridge Base Table	316
12.22.2	SRP Bridge Port Table	316
12.22.3	SRP Latency Parameter Table	316
12.22.4	SRP Stream Table	317
12.22.5	SRP Reservations Table	317
13.	Spanning Tree Protocols	319
13.1	Protocol design requirements.....	320
13.2	Protocol support requirements	321
13.2.1	MSTP support requirements	321
13.3	Protocol design goals	321
13.4	RSTP overview	322
13.4.1	Computation of the active topology	322
13.4.2	Example topologies	324
13.5	MSTP overview	327
13.5.1	Example topologies	328
13.5.2	Relationship of MSTP to RSTP	331
13.5.3	Modeling an MST Region as a single bridge	331
13.6	Compatibility and interoperability.....	332
13.6.1	Designated Port selection	332
13.6.2	Force Protocol Version	332
13.7	MST Configuration Identifier	333
13.8	Spanning Tree Priority Vectors	334
13.9	CIST Priority Vector calculations	336
13.10	MST Priority Vector calculations	337
13.11	Port Role assignments.....	339
13.12	Stable connectivity	340
13.13	Communicating Spanning Tree information	341
13.14	Changing Spanning Tree information.....	342
13.15	Changing Port States with RSTP or MSTP	343
13.15.1	Subtree connectivity and priority vectors	343
13.15.2	Root Port transition to Forwarding	343
13.15.3	Designated Port transition to Forwarding	344

13.15.4	Master Port transition to Forwarding	344
13.16	Managing spanning tree topologies	347
13.17	Updating learned station location information	349
13.18	Managing reconfiguration.....	350
13.19	Partial and disputed connectivity	351
13.20	In-service upgrades	351
13.21	Fragile bridges	352
13.22	Spanning tree protocol state machines.....	353
13.23	State machine timers	355
13.23.1	edgeDelayWhile	356
13.23.2	fdWhile	356
13.23.3	helloWhen	356
13.23.4	mdelayWhile	356
13.23.5	rbWhile	356
13.23.6	rvdInfoWhile	356
13.23.7	rrWhile	357
13.23.8	tcDetected	357
13.23.9	tcWhile	357
13.23.10	pseudoInfoHelloWhen	357
13.24	Per bridge variables	357
13.24.1	BridgeIdentifier	358
13.24.2	BridgePriority	358
13.24.3	BridgeTimes	358
13.24.4	ForceProtocolVersion	358
13.24.5	MigrateTime	358
13.24.6	MstConfigId	359
13.24.7	rootPortId	359
13.24.8	rootPriority	359
13.24.9	rootTimes	359
13.24.10	TxHoldCount	359
13.25	Per port variables	359
13.25.1	AdminEdge	361
13.25.2	ageingTime	361
13.25.3	agree	361
13.25.4	agreed	361
13.25.5	AutoEdge	361
13.25.6	AutoIsolate	361
13.25.7	designatedPriority	362
13.25.8	designatedTimes	362
13.25.9	disputed	362
13.25.10	enableBPDUrx	362
13.25.11	enableBPDUtx	362
13.25.12	ExternalPortPathCost	362
13.25.13	fdbFlush	362
13.25.14	forward	363
13.25.15	forwarding	363
13.25.16	infoInternal	363
13.25.17	infoIs	363
13.25.18	InternalPortPathCost	363
13.25.19	isL2gp	363
13.25.20	isolate	363
13.25.21	learn	363
13.25.22	learning	364
13.25.23	master	364

13.25.24	mastered	364
13.25.25	mcheck	364
13.25.26	msgPriority	364
13.25.27	msgTimes	364
13.25.28	newInfo	364
13.25.29	newInfoMsti	365
13.25.30	operEdge	365
13.25.31	portEnabled	365
13.25.32	portId	365
13.25.33	portPriority	365
13.25.34	portTimes	365
13.25.35	proposed	365
13.25.36	proposing	365
13.25.37	pseudoRootId	366
13.25.38	rvdBPDU	366
13.25.39	rvdInfo	366
13.25.40	rvdInternal	366
13.25.41	rvdMsg	366
13.25.42	rvdRSTP	366
13.25.43	rvdSTP	366
13.25.44	rvdTc	366
13.25.45	rvdTcAck	366
13.25.46	rvdTcn	366
13.25.47	reRoot	366
13.25.48	reselect	366
13.25.49	restrictedRole	367
13.25.50	restrictedTcn	367
13.25.51	role	367
13.25.52	selected	367
13.25.53	selectedRole	367
13.25.54	sendRSTP	367
13.25.55	sync	367
13.25.56	synced	367
13.25.57	tcAck	367
13.25.58	tcProp	368
13.25.59	tick	368
13.25.60	txCount	368
13.25.61	updInfo	368
13.26	State machine conditions and parameters	368
13.26.1	allSynced	368
13.26.2	allTransmitReady	369
13.26.3	cist	369
13.26.4	cistRootPort	369
13.26.5	cistDesignatedPort	369
13.26.6	EdgeDelay	369
13.26.7	forwardDelay	369
13.26.8	FwdDelay	369
13.26.9	HelloTime	369
13.26.10	msti	369
13.26.11	MaxAge	369
13.26.12	mstiDesignatedOrTCpropagatingRootPort	369
13.26.13	mstiMasterPort	370
13.26.14	operPointToPoint	370
13.26.15	rvdAnyMsg	370

13.26.16	rcvdCistMsg	370
13.26.17	rcvdMstiMsg	370
13.26.18	reRooted	370
13.26.19	rstpVersion	370
13.26.20	stpVersion	370
13.26.21	updCistInfo	370
13.26.22	updMstiInfo	370
13.27	State machine procedures	370
13.27.1	betterOrSameInfo(newInfoIs)	371
13.27.2	clearAllRcvdMsgs()	372
13.27.3	clearReselectTree()	372
13.27.4	disableForwarding()	372
13.27.5	disableLearning()	372
13.27.6	enableForwarding()	372
13.27.7	enableLearning()	372
13.27.8	fromSameRegion()	372
13.27.9	newTcDetected()	372
13.27.10	newTcWhile()	372
13.27.11	pseudoRcvMsgs()	373
13.27.12	rcvInfo()	373
13.27.13	rcvMsgs()	374
13.27.14	recordAgreement()	374
13.27.15	recordDispute()	374
13.27.16	recordMastered()	375
13.27.17	recordPriority()	375
13.27.18	recordProposal()	375
13.27.19	recordTimes()	375
13.27.20	setReRootTree()	375
13.27.21	setSelectedTree()	375
13.27.22	setSyncTree()	375
13.27.23	setTcFlags()	375
13.27.24	setTcPropTree()	376
13.27.25	syncMaster()	376
13.27.26	txConfig()	376
13.27.27	txRstp()	376
13.27.28	txTcn()	377
13.27.29	updBPDUVersion()	377
13.27.30	updRcvdInfoWhile()	377
13.27.31	updRolesTree()	377
13.27.32	uptRolesDisabledTree()	379
13.28	The Port Timers state machine.....	379
13.29	Port Receive state machine	379
13.30	Port Protocol Migration state machine	380
13.31	Bridge Detection state machine	380
13.32	Port Transmit state machine.....	381
13.33	Port Information state machine.....	382
13.34	Port Role Selection state machine	383
13.35	Port Role Transitions state machine	383
13.36	Port State Transition state machine.....	387
13.36.1	Port State transitions for the CIST and MSTIs	387
13.37	Topology Change state machine	388
13.38	Layer 2 Gateway Port Receive state machine	389
13.39	Customer Edge Port Spanning Tree operation.....	389
13.39.1	Provider Edge Port operPointToPointMAC and operEdge	389

13.39.2	updRolesTree()	390
13.39.3	setReRootTree(), setSyncTree(), setTcPropTree()	390
13.39.4	allSynced, reRooted	390
13.39.5	Configuration parameters	390
13.40	Virtual Instance Port Spanning Tree operation	391
14.	Use of BPDUs by MSTP	392
14.1	BPDU Structure	392
14.1.1	Transmission and representation of octets	392
14.1.2	Components	392
14.2	Encoding of parameter types	392
14.2.1	Encoding of Port Role values	392
14.2.2	Allocation and encoding of Bridge Identifiers	393
14.2.3	Allocation and encoding of Port Identifiers	393
14.2.4	Encoding of External Root Path Cost	393
14.2.5	Encoding of Internal Root Path Cost	393
14.2.6	Encoding of Hop Counts	393
14.3	BPDU formats and parameters	394
14.3.1	STP BPDUs	394
14.3.2	RST BPDUs	394
14.3.3	MST BPDUs	394
14.4	Validation of received BPDUs	394
14.5	Transmission of BPDUs	395
14.6	Encoding and decoding of STP Configuration, RST, and MST BPDUs	396
14.6.1	MSTI Configuration Messages	397
15.	Support of the MAC Service by Provider Bridged Networks	398
15.1	Service transparency	398
15.2	Customer service interfaces	399
15.3	Port-based service interface	399
15.4	C-tagged service interface	400
15.5	S-tagged service interface	401
15.6	Service instance segregation	402
15.7	Service instance selection and identification	402
15.8	Service priority selection	403
15.9	Service access protection	403
15.10	Connectivity Fault Management	404
15.11	Data-driven and data-dependent connectivity fault management (DDCFM)	404
16.	Principles of Provider Bridged Network operation	405
16.1	Provider Bridged Network overview	405
16.2	Provider Bridged Network	406
16.3	Service instance connectivity	407
16.4	Service provider learning of customer end station addresses	408
16.5	Detection of connectivity loops through attached networks	408
16.6	Network management	409
17.	Management Information Base (MIB)	410
17.1	Internet Standard Management Framework	410
17.2	Structure of the MIB	410
17.2.1	Structure of the IEEE8021-TC MIB	411
17.2.2	Structure of the IEEE8021-BRIDGE MIB	412

17.2.3	Structure of the IEEE8021-SPANNING-TREE MIB	417
17.2.4	Structure of the IEEE8021-Q-BRIDGE MIB	419
17.2.5	Structure of the IEEE8021-PB MIB	426
17.2.6	Structure of the IEEE8021-MSTP MIB	428
17.2.7	Structure of the IEEE8021-CFM MMIB	431
17.2.8	Structure of the IEEE8021-PBB MIB	437
17.2.9	Structure of the IEEE8021-DDCFM MIBs	440
17.2.10	Structure of the IEEE8021-PBBTE MIB	442
17.2.11	Structure of the TPMR MIB	445
17.2.12	Structure of the IEEE8021-FQTSS MIB	447
17.2.13	Structure of the Congestion Notification MIB	448
17.2.14	Structure of the IEEE8021-SRP MIB	450
17.3	Relationship to other MIBs.....	452
17.3.1	Relationship of the IEEE8021-TC MIB to other MIB modules	452
17.3.2	Relationship of the IEEE8021-BRIDGE MIB to other MIB modules	452
17.3.3	Relationship of the IEEE8021-RSTP MIB to other MIB modules	455
17.3.4	Relationship of the IEEE8021-Q-BRIDGE MIB to other MIB modules	455
17.3.5	Relationship of the IEEE8021-PB-BRIDGE MIB to other MIB modules	456
17.3.6	Relationship of the IEEE8021-MSTP MIB to other MIB modules	457
17.3.7	Relationship of the IEEE8021-CFM MIB to other MIB modules	457
17.3.8	Relationship of the IEEE8021-PBB MIB to other MIB modules	458
17.3.9	Relationship of the IEEE8021-DDCFM to other MIB modules	459
17.3.10	Relationship of the IEEE8021-PBBTE MIB to other MIB modules	459
17.3.11	Relationship of the TPMR MIB to other MIB modules	460
17.3.12	Relationship of the IEEE8021-FQTSS MIB to other MIB modules	460
17.3.14	Relationship of the IEEE8021-SRP MIB to other MIB modules	461
17.4	Security considerations	461
17.4.1	Security considerations of the IEEE8021-TC MIB	461
17.4.2	Security considerations of the IEEE8021-BRIDGE MIB	461
17.4.3	Security considerations of the IEEE8021-SPANNING-TREE MIB	462
17.4.4	Security considerations of the IEEE8021-Q-BRIDGE MIB	463
17.4.5	Security considerations of the IEEE8021-PB MIB	464
17.4.6	Security considerations of the IEEE8021-MSTP MIB	464
17.4.7	Security considerations of the IEEE8021-CFM MIB	465
17.4.8	Security considerations of the IEEE8021-PBB MIB	467
17.4.9	Security considerations of the IEEE8021-DDCFM MIB	467
17.4.10	Security considerations of the IEEE8021-PBBTE MIB	468
17.4.11	Security considerations of the TPMR MIB	469
17.4.12	Security considerations of the IEEE8021-FQTSS MIB	469
17.4.13	Security considerations of the Congestion Notification MIB	470
17.4.14	Security considerations of the IEEE8021-SRP MIB	471
17.5	Dynamic component and Port creation.....	472
17.5.1	Overview of the dynamically created Bridge entities	472
17.5.2	Component creation	473
17.5.3	Port creation	474
17.6	MIB operations for service interface configuration.....	481
17.6.1	Provisioning Provider Bridged Network service interfaces	481
17.6.2	Provisioning Backbone Bridged Network service interfaces	484
17.7	MIB modules	490
17.7.1	Definitions for the IEEE8021-TC MIB module	490
17.7.2	Definitions for the IEEE8021-BRIDGE MIB module	499
17.7.3	Definitions for the IEEE8021-SPANNING-TREE MIB module	534
17.7.4	Definitions for the IEEE8021-Q-BRIDGE MIB module	551
17.7.5	Definitions for the IEEE8021-PB MIB module	591

17.7.6	Definitions for the IEEE8021-MSTP MIB module	604
17.7.7	Definitions for the IEEE8021-CFM MIB module	629
17.7.8	Definitions for the IEEE8021-PBB MIB module	713
17.7.9	Definitions for the IEEE8021-DDCFM MIB module	735
17.7.10	Definitions for the IEEE8021-PBBTE MIB module	752
17.7.11	Definitions for the TPMR MIB module	769
17.7.12	Definitions for the IEEE8021-FQTSS MIB module	783
17.7.13	Congestion Notification MIB module	794
17.7.14	Definitions of the IEEE8021-SRP MIB module	828
18.	Principles of Connectivity Fault Management operation	844
18.1	Maintenance Domains and Domain Service Access Points	845
18.2	Service instances and Maintenance Associations	847
18.3	Maintenance Domain Levels	848
19.	Connectivity Fault Management Entity operation.....	852
19.1	Maintenance Points.....	852
19.2	Maintenance association End Point.....	852
19.2.1	MEP identification	853
19.2.2	MEP functions	854
19.2.3	MEP architecture	854
19.2.4	MP Type Demultiplexer	854
19.2.5	MP Multiplexer	856
19.2.6	MP Level Demultiplexer	856
19.2.7	MP OpCode Demultiplexer	856
19.2.8	MEP Continuity Check Receiver	856
19.2.9	MEP Continuity Check Initiator	857
19.2.10	MP Loopback Responder	857
19.2.11	MEP Loopback Initiator	857
19.2.12	MEP Linktrace Initiator	857
19.2.13	MEP LTI SAP	858
19.2.14	MEP Linktrace SAP	858
19.2.15	MEP CCM Database	858
19.2.16	MEP Fault Notification Generator	858
19.2.17	MEP Decapsulator Responder	858
19.2.18	MEP RFM Receiver	858
19.3	MIP Half Function	859
19.3.1	MHF identification	859
19.3.2	MHF functions	859
19.3.3	MHF architecture	860
19.3.4	MHF Level Demultiplexer	860
19.3.5	MHF Type Demultiplexer	861
19.3.6	MHF OpCode Demultiplexer	861
19.3.7	MHF Multiplexer	861
19.3.8	MHF Loopback Responder	861
19.3.9	MHF Continuity Check Receiver	861
19.3.10	MIP CCM Database	861
19.3.11	MHF Linktrace SAP	861
19.3.12	MHF Decapsulator Responder	861
19.3.13	MHF RFM Receiver	861
19.4	Maintenance Point addressing	862
19.5	Linktrace Output Multiplexer	862
19.6	Linktrace Responder	863

20. Connectivity Fault Management protocols.....	865
20.1 Continuity Check protocol.....	866
20.2 Loopback protocol	869
20.3 Linktrace protocol.....	871
20.4 Connectivity Fault Management state machines	874
20.5 CFM state machine timers	876
20.6 CFM procedures	877
20.7 Maintenance Domain variable	877
20.8 Maintenance Association variables	878
20.9 MEP variables.....	878
20.9.8 presentTraffic	880
20.9.9 presentmmLoc	880
20.10 MEP Continuity Check Initiator variables.....	880
20.11 MEP Continuity Check Initiator procedures	881
20.12 MEP Continuity Check Initiator state machine	881
20.13 MHF Continuity Check Receiver variables.....	882
20.14 MHF Continuity Check Receiver procedures.....	882
20.15 MHF Continuity Check Receiver state machine	882
20.16 MEP Continuity Check Receiver variables	883
20.17 MEP Continuity Check Receiver procedures	885
20.18 MEP Continuity Check Receiver state machine.....	886
20.19 Remote MEP variables	886
20.20 Remote MEP state machine	888
20.21 Remote MEP Error variables.....	888
20.22 Remote MEP Error state machine	889
20.23 MEP Cross Connect variables	889
20.24 MEP Cross Connect state machine.....	890
20.25 MEP Mismatch variables.....	890
20.25.1 mmCCMReceived	891
20.25.2 mmCCMdefect	891
20.25.3 mmCCMTime	891
20.25.4 disableLocdefect	891
20.25.5 mmLocdefect	891
20.26 MEP Mismatch state machines.....	891
20.27 MP Loopback Responder variables	891
20.28 MP Loopback Responder procedures	893
20.29 MP Loopback Responder state machine.....	894
20.30 MEP Loopback Initiator variables	894
20.31 MEP Loopback Initiator transmit procedures.....	895
20.32 MEP Loopback Initiator transmit state machine	896
20.33 MEP Loopback Initiator receive procedures	897
20.34 MEP Loopback Initiator receive state machine	897
20.35 MEP Fault Notification Generator variables	898
20.36 MEP Fault Notification Generator procedures	899
20.37 MEP Fault Notification Generator state machine	899
20.38 MEP Mismatch Fault Notification Generator variables	899
20.39 MEP Mismatch Fault Notification Generator procedures	901
20.40 MEP Mismatch Fault Notification Generator state machine	901
20.41 MEP Linktrace Initiator variables.....	902
20.42 MEP Linktrace Initiator procedures	904
20.43 MEP Linktrace Initiator receive variables	905
20.44 MEP Linktrace Initiator receive procedures	905
20.45 MEP Linktrace Initiator receive state machine	905
20.46 Linktrace Responder variables.....	906

20.47	LTM Receiver procedures	906
20.48	LTM Receiver state machine	912
20.49	LTR Transmitter procedure	913
20.50	LTR Transmitter state machine	913
20.51	CFM PDU validation and versioning	914
20.52	PDU identification	917
20.53	Use of transaction IDs and sequence numbers	917
21.	Encoding of CFM Protocol Data Units.....	919
21.1	Structure, representation, and encoding.....	919
21.2	CFM encapsulation	919
21.3	CFM request and indication parameters	920
21.4	Common CFM Header.....	921
21.5	TLV Format	922
21.6	Continuity Check Message format	927
21.7	Loopback Message and Loopback Reply formats	932
21.7.4	Additional LBM/LBR TLVs	933
21.7.5	PBB-TE MIP TLV	933
21.8	Linktrace Message Format.....	934
21.9	Linktrace Reply Format	936
22.	Connectivity Fault Management in systems	942
22.1	CFM shims in Bridges	942
22.1.1	Preliminary positioning of Maintenance Points	942
22.1.2	CFM and the Forwarding Process	943
22.1.3	Up/Down separation of Maintenance Points	944
22.1.4	Service instances over multiple Bridges	947
22.1.5	Multiple VID service instances	949
22.1.6	Untagged CFM PDUs	949
22.1.7	Maintenance Points and non-VLAN aware Bridges	949
22.1.8	Maintenance Points and other standards	950
22.1.9	CFM and IEEE 802.3 Clause 57 OAM	952
22.2	Maintenance Entity creation	952
22.2.1	Creating Maintenance Domains and Maintenance Associations	953
22.2.2	Creating MEPs	953
22.2.3	Creating MIPs	955
22.2.4	CFM configuration errors	956
22.3	MPs, Ports, and MD Level assignment.....	957
22.4	Stations and Connectivity Fault Management.....	957
22.5	Scalability of Connectivity Fault Management	958
22.6	CFM in Provider Bridges.....	959
22.6.1	Maintenance Points and C-VLAN components	959
22.6.2	Maintenance C-VLAN on a Port-based service interface	960
22.6.3	Maintenance C-VLAN on a C-tagged service interface	961
22.7	Management Port MEPs and CFM in the enterprise environment.....	961
22.8	Implementing CFM on existing Bridges	962
23.	MAC status propagation	965
23.1	Model of operation.....	966
23.1.1	MAC Status Shim	967
23.1.2	Relationship of Connectivity Fault Management to the MAC Status Shim	968
23.2	MAC status protocol (MSP) overview	968
23.3	MAC status protocol state machines	972
23.4	State machine timers	974

23.4.1	linkNotifyWhen	974
23.4.2	linkNotifyWhile	974
23.4.3	macNotifyWhile	974
23.4.4	macRecoverWhile	974
23.5	MSP performance parameters.....	974
23.5.1	LinkNotify	974
23.5.2	LinkNotifyWait	975
23.5.3	LinkNotifyRetry	975
23.5.4	MACNotify	975
23.5.5	MACNotifyTime	975
23.5.6	MACRecoverTime	975
23.6	State machine variables	975
23.6.1	BEGIN	975
23.6.2	addConfirmed	975
23.6.3	disableMAC	975
23.6.4	disabledMAC	975
23.6.5	disableMSS	975
23.6.6	lossConfirmed	975
23.6.7	macOperational	976
23.6.8	mssOperational	976
23.6.9	prop	976
23.6.10	rxAck	976
23.6.11	rxAdd	976
23.6.12	rxAddConfirm	976
23.6.13	rxLoss	976
23.6.14	rxLossConfirm	976
23.6.15	txAck	976
23.6.16	txAdd	976
23.6.17	txAddConfirm	976
23.6.18	txLoss	976
23.6.19	txLossConfirm	977
23.7	State machine procedures	977
23.8	Status Transition state machine	977
23.9	Status Notification state machine	977
23.10	Receive Process	977
23.11	Transmit Process.....	978
23.12	Management of MSP	978
23.13	MSPDU transmission, addressing, and protocol identification.....	979
23.13.1	Destination MAC Address	979
23.13.2	Source MAC Address	979
23.13.3	Priority	979
23.13.4	EtherType use and encoding	980
23.14	Representation and encoding of octets	980
23.15	MSPDU structure.....	980
23.15.1	Protocol Version	980
23.15.2	Packet Type	981
23.16	Validation of received MSPDUs	981
23.17	Other MSP participants.....	981
24.	<Reserved for a future amendment>	982
25.	Support of the MAC Service by Provider Backbone Bridged Networks	983
25.1	Service transparency	985

25.2	Customer service interface.....	985
25.3	Port-based service interface	986
25.4	S-tagged service interface.....	987
25.5	I-tagged service interface.....	989
25.6	Service instance segregation.....	991
25.7	Service instance selection and identification	991
25.8	Service priority and drop eligibility selection.....	992
25.9	Service access protection	992
25.9.1	Class II redundant LANs access protection	994
25.9.2	Class III simple redundant LANs and nodes access protection	995
25.10	Support of the MAC Service by a PBB-TE Region	996
25.10.1	Provisioning TESIs	997
25.10.2	ESP forwarding behavior	998
25.11	Transparent service interface	999
26.	Principles of Provider Backbone Bridged Network operation	1001
26.1	Provider Backbone Bridged Network overview	1001
26.2	Provider Backbone Bridged Network example	1002
26.3	Backbone VLAN connectivity.....	1004
26.4	Backbone addressing	1005
26.4.1	Learning individual backbone addresses at a PIP	1005
26.4.2	Translating backbone destination addresses at a CBP	1006
26.4.3	Backbone addressing considerations for CFM Maintenance Points	1006
26.5	Detection of connectivity loops through attached networks.....	1007
26.6	Scaling of Provider Backbone Bridges	1007
26.6.1	Hierarchal PBBNs	1007
26.6.2	Peer PBBNs	1008
26.7	Network Management.....	1008
26.8	Connectivity Fault Management in Provider Backbone Bridges	1008
26.8.1	CFM over Port-based and S-tagged Service Interfaces	1014
26.8.2	Connectivity Fault Management over I-tagged Service Interfaces	1015
26.8.3	Connectivity Fault Management over hierachal E-NNI	1015
26.8.4	Connectivity Fault Management over peer E-NNI	1015
26.9	Connectivity Fault Management in a PBB-TE Region	1016
26.9.1	Addressing PBB-TE MEPs	1016
26.9.2	TESI identification	1016
26.9.3	PBB-TE MEP placement in a Bridge Port	1016
26.9.4	PBB-TE MIP placement in a Bridge Port	1018
26.9.5	TESI Maintenance Domains	1018
26.10	Protection switching for point-to-point TESIs.....	1020
26.10.1	Introduction	1020
26.10.2	1:1 point-to-point TESI protection switching	1021
26.10.3	Protection Switching state machines	1024
26.11	Mismatch defect.....	1029
27.	<Reserved for a future amendment>	1031
28.	<Reserved for a future amendment>	1032
29.	DDCFM operations and protocols.....	1033
29.1	Principles of DDCFM operation.....	1033
29.1.1	Data-driven and data-dependent faults (DDFs)	1033
29.1.2	Basic principle to diagnose and isolate DDFs	1033

29.2	DDCFM Entity operation	1036
29.2.1	DDCFM implementation	1036
29.2.2	Forward Path Test Reflection Responder	1037
29.2.3	Reflection Responder related parameters	1038
29.2.4	Reflection Target and RFM Receiver	1039
29.2.5	Return path test related parameters	1039
29.2.6	Decapsulator Responder	1040
29.2.7	SFM Originator	1041
29.3	DDCFM protocols	1041
29.3.1	Reflection Responder variables	1041
29.3.2	RR Filter Procedures	1043
29.3.3	RR Encapsulation Procedures	1044
29.3.4	RR Transmit procedure	1045
29.3.5	Reflection Responder related state machines	1046
29.3.6	RFM Receiver variables	1047
29.3.7	RFM Receiver procedure	1048
29.3.8	Decapsulator Responder variables	1049
29.3.9	Decapsulator Responder procedures	1049
29.3.10	Decapsulator Responder state machine	1051
29.4	Encoding of DDCFM Protocol Data Units.....	1051
29.4.1	RFM and SFM Header	1051
29.4.2	RFM format	1051
29.4.3	SFM format	1052
30.	Principles of congestion notification	1054
30.1	Congestion notification design requirements	1054
30.2	Quantized Congestion Notification protocol	1056
30.3	Congestion Controlled Flow	1060
30.4	Congestion Notification Priority Value	1061
30.5	Congestion Notification Tag.....	1061
30.6	Congestion Notification Domain	1061
30.7	Multicast data.....	1062
30.8	Congestion notification and additional tags.....	1063
31.	Congestion notification entity operation.....	1064
31.1	Congestion aware Bridge Forwarding Process	1064
31.2	Congestion aware end station functions	1065
32.	Congestion notification protocol	1071
32.1	Congestion Notification Domain operations	1071
32.2	CN component variables.....	1074
32.3	Congestion notification per-CNPV variables	1075
32.4	CND defense per-Port per-CNPV variables	1077
32.5	Congestion Notification Domain defense procedures	1080
32.6	Congestion Notification Domain defense state machine	1081
32.7	Congestion notification protocol	1081
32.8	Congestion Point variables	1082
32.9	Congestion Point procedures	1084
32.10	Reaction Point per-Port per-CNPV variables	1087
32.11	Reaction Point group variables	1087
32.12	Reaction Point timer	1089
32.13	Reaction Point variables	1089
32.14	Reaction Point procedures	1090
32.15	RP rate control state machine	1092
32.16	Congestion notification and encapsulation interworking function	1094

33. Encoding of congestion notification Protocol Data Units	1096
33.1 Structure, representation, and encoding.....	1096
33.2 Congestion Notification Tag format.....	1096
33.3 Congestion Notification Message.....	1097
33.4 Congestion Notification Message PDU format	1098
34. Forwarding and queuing for time-sensitive streams.....	1101
34.1 Overview.....	1101
34.2 Detection of SRP domains	1101
34.3 The bandwidth availability parameters.....	1102
34.3.1 Relationships among bandwidth availability parameters	1102
34.3.2 Bandwidth availability parameter management	1103
34.4 Deriving actual bandwidth requirements from the size of the MSDU	1103
34.5 Mapping priorities to traffic classes for time-sensitive streams	1104
34.6 End station behavior	1106
34.6.1 Talker behavior	1106
34.6.2 Listener behavior	1107
35. Stream Registration Protocol (SRP)	1108
35.1 Multiple Stream Registration Protocol (MSRP).....	1109
35.1.1 MSRP and Shared Media	1110
35.1.2 Behavior of end stations	1110
35.1.3 Behavior of Bridges	1112
35.2 Definition of the MSRP application	1112
35.2.1 Definition of internal state variables	1112
35.2.2 Definition of MRP elements	1114
35.2.3 Provision and support of Stream registration service	1125
35.2.4 MSRP Attribute Propagation	1129
35.2.5 Operational reporting and statistics	1133
35.2.6 Encoding	1134
35.2.7 Attribute value support requirements	1134
Annex A (normative) PICS proforma—Bridge implementations	1135
Annex B (normative) PICS proforma—End station implementations	1185
Annex C (normative) DMN (Designated MSRP Node) Implementations	1197
Annex D (normative) IEEE 802.1 Organizational Specific TLVs	1214
Annex E (normative) Notational conventions used in state diagrams	1258
Annex F (informative) Shared and Independent VLAN Learning	1260
Annex G (informative) MAC method dependent aspects of VLAN support	1269
Annex H (informative) Interoperability considerations.....	1271
Annex I (informative) Priority and drop precedence.....	1279
Annex J (informative) Connectivity Fault Management protocol design and use	1287
Annex K (informative) TPMR use cases	1295

Annex L (informative) Operation of the credit-based shaper algorithm	1300
Annex M (informative) Bibliography	1317

Figures

Figure 6-1—Internal organization of the MAC sublayer	47
Figure 6-2—MAC entities, the MAC Service, and MAC Service users (clients).....	49
Figure 6-3—An interface stack.....	50
Figure 6-4—Provider Instance Ports	74
Figure 6-5—B-Component Customer Backbone Port.....	77
Figure 6-6—Example of operation of Port-and-Protocol-based classification.....	81
Figure 6-7—Service access priority selection	84
Figure 6-8—Two back-to-back EISS Multiplex Entities	90
Figure 6-9—Two back-to-back Backbone Service Instance Multiplex Entities	91
Figure 6-10—Backbone Service Instance Multiplex Entities with example CFM shims	91
Figure 6-11—Two back-to-back Up and Down TESI Multiplex Entities.....	94
Figure 6-12—Supporting the ISS with signaled priority	95
Figure 7-1—VLAN Bridging overview	98
Figure 8-1—A Bridged Local Area Network	104
Figure 8-2—VLAN-aware Bridge architecture	106
Figure 8-3—Relaying MAC frames	108
Figure 8-4—Observation of network traffic	108
Figure 8-5—Operation of Spanning Tree protocol.....	109
Figure 8-6—Operation of MRP	109
Figure 8-7—Management Port transmission and reception	110
Figure 8-8—Bridge Port Transmit and Receive	112
Figure 8-9—TPMR Port Transmit and Receive	112
Figure 8-10—Forwarding Process functions	114
Figure 8-11—Logical points of attachment of the Higher Layer and Relay Entities.....	142
Figure 8-12—Effect of control information on the forwarding path.....	142
Figure 8-13—Per-Port points of attachment.....	143
Figure 8-14—Single point of attachment—relay permitted	143
Figure 8-15—Single point of attachment—relay not permitted	144
Figure 8-16—Effect of Port State	144
Figure 8-17—Controlled and Uncontrolled Port connectivity	145
Figure 8-18—Ingress/egress control information in the forwarding path	145
Figure 9-1—VLAN TCI format	150
Figure 9-3—I-TAG TCI format.....	151
Figure 10-1—Example—Attribute value propagation from one station	154
Figure 10-2—Example—Attribute value propagation from two stations	154
Figure 10-3—Example—Registrations as pointers to the sources of declarations	155
Figure 10-4—MRP architecture	157
Figure 10-5—Format of the major components of an MRPDU	180
Figure 10-6—Operation of MMRP for a single VLAN Context.....	186
Figure 10-7—Example Directed Graph.....	187
Figure 10-8—Example of MMRP propagation in a VLAN Context	189
Figure 11-1—Operation of MVRP	197
Figure 12-1—Relationships among CFM managed objects.....	256
Figure 12-2—Relationship among BEB managed objects	272
Figure 13-1—Diagrammatic conventions for spanning tree topologies.....	324
Figure 13-2—Physical topology and active topology	325
Figure 13-3—Port Roles and Port States	325
Figure 13-4—A Backup Port	326
Figure 13-5—“Ring Backbone” example.....	326
Figure 13-6—An MST Bridge network.....	328
Figure 13-7—CIST Priority Vectors, Port Roles, and MST Regions	329

Figure 13-8—MSTI Active Topology in Region 2	330
Figure 13-9—CIST and MSTI active topologies in Region 1 of the example network	341
Figure 13-10—Agreements and Proposals	345
Figure 13-11—CIST and MSTI Active Topologies in Region 2 of Figure 13-6—	346
Figure 13-12—Enhanced Agreements.....	347
Figure 13-13—Spanning tree protocol state machines—overview and relationships.....	354
Figure 13-14—MSTP overview notation	355
Figure 13-15—Port Timers state machine.....	379
Figure 13-16—Port Receive state machine	379
Figure 13-17—Port Protocol Migration state machine.....	380
Figure 13-18—Bridge Detection state machine	380
Figure 13-19—Port Transmit state machine.....	381
Figure 13-20—Port Information state machine	382
Figure 13-21—Port Role Selection state machine.....	383
Figure 13-22—Disabled Port role transitions	384
Figure 13-23—Port Role Transitions state machine—MasterPort.....	384
Figure 13-24—Port Role Transitions state machine—RootPort	385
Figure 13-25—Port Role Transitions state machine—DesignatedPort	386
Figure 13-26—Port Role Transitions state machine—AlternatePort and BackupPort	386
Figure 13-27—Port State Transition state machine.....	387
Figure 13-28—Topology Change state machine	388
Figure 13-29—L2 Gateway Port Receive state machine.....	389
Figure 14-1—MST BPDU parameters and format.....	394
Figure 14-2—MSTI Configuration Message parameters and format.....	397
Figure 15-1—Internal organization of the MAC sublayer in a Provider Bridged Network.....	398
Figure 15-2—Port-based service interface to a Provider Bridged Network	399
Figure 15-3—Port-based service interface to a Provider Bridged Network.....	400
Figure 15-4—C-tagged service interface to a Provider Bridged Network	400
Figure 15-5—C-tagged service interface to a Provider Bridged Network	400
Figure 15-6—Customer Edge Ports.....	401
Figure 15-7—S-tagged service interface to a Provider Bridged Network.....	401
Figure 15-8—S-tagged interface to a Provider Bridged Network	402
Figure 16-1—Provider Bridged Network with interface examples	406
Figure 17-1— C-VLAN component internal LAN managed system.....	454
Figure 17-2—I/B-component internal LAN managed system	459
Figure 18-1—One Maintenance Domain: operator’s view	846
Figure 18-2—One service instance: operator’s view.....	847
Figure 18-3—One service instance: customer’s view	847
Figure 18-4—MEP and MIP Symbols.....	848
Figure 18-5—Maintenance Associations: one service instance in a provider network.....	849
Figure 18-6—Maintenance Associations: Expansion of Figure 18-5	850
Figure 18-7—MEPs, MIPs, and MD Levels	851
Figure 19-1—CFM Protocol shims	852
Figure 19-2—Maintenance association End Point (MEP)	855
Figure 19-3—MIP Half Function	860
Figure 19-4—Linktrace Output Multiplexer shim.....	862
Figure 19-5—Linktrace Output Multiplexer architecture	863
Figure 20-1—MEP state machines—overview and relationships	875
Figure 20-2—MEP Continuity Check Initiator state machine	882
Figure 20-3—MHF Continuity Check Receiver state machine.....	883
Figure 20-4—MEP Continuity Check Receiver state machine	886
Figure 20-5—Remote MEP state machine	888
Figure 20-6—Remote MEP Error state machine.....	889
Figure 20-7—MEP Cross Connect state machine	890

Figure 20-8—MEP Traffic Field Mismatch state machine	892
Figure 20-9—MEP Local Mismatch state machine	892
Figure 20-10—MP Loopback Responder state machine	894
Figure 20-11—MEP Loopback Initiator transmit state machine	896
Figure 20-12—MEP Loopback Initiator receive state machine	897
Figure 20-13—MEP Fault Notification Generator state machine	900
Figure 20-14—MEP Mismatch Fault Notification Generator state machine	901
Figure 20-15—MEP Linktrace Initiator receive state machine	906
Figure 20-16—Linktrace Responder, MEPs, MHFs, and LOMs	907
Figure 20-17—LTM Receiver state machine	913
Figure 20-18—LTR Transmitter state machine	913
Figure 22-1—MEPs and MIPs distinguished by VID (incomplete picture)	943
Figure 22-2—Alternate view of Forwarding process	944
Figure 22-3—Combining per-VLAN Maintenance Points into two shims	945
Figure 22-4—More complete picture of Maintenance Point placement in a Bridge Port	946
Figure 22-5—Service instance spanning two Bridges protected by Up MPs	948
Figure 22-6—Service instance spanning two Bridges protected by Down MPs	948
Figure 22-7—Maintenance Point placement in a non-VLAN aware Bridge Port	950
Figure 22-8—Maintenance Point placement relative to other standards	951
Figure 22-9—Creating MEPs and MIPs	954
Figure 22-10—CFM in a Provider Edge Bridge C-tagged service interface	960
Figure 22-11—Up MEPs in a Management Port	961
Figure 22-12—CFM in the enterprise environment	963
Figure 22-13—CFM on a Bridge conformant to IEEE Std 802.1Q-2005	963
Figure 23-1—TPMR connecting two Bridge Ports	965
Figure 23-2—TPMR chain connecting Bridge Ports	965
Figure 23-3—MAC Status Shims and the MAC Status Propagation Entity	967
Figure 23-4—Adding connectivity	969
Figure 23-5—Losing connectivity	970
Figure 23-6—TPMR recovery	971
Figure 23-7—Notification from one end of the link to the other	972
Figure 23-8—Immediate MAC status notification at the end of a link	972
Figure 23-9—MSPE state machine overview	973
Figure 23-10—Status Transition state machine	977
Figure 23-11—Status Notification state machine	978
Figure 23-12—MSPDU structure	980
Figure 25-1—Internal organization of the MAC sublayer in a PBBN	983
Figure 25-2—Provider Backbone Bridge terminology	984
Figure 25-3—Customer service interface types	985
Figure 25-4—Port-based service interface	986
Figure 25-5—Port-based interface equipment	987
Figure 25-6—Encapsulated service frames at ISS	988
Figure 25-7—S-tagged service interface	988
Figure 25-8—S-tagged service interface equipment	989
Figure 25-9—I-tagged service interface	990
Figure 25-10—I-tagged service interface equipment	990
Figure 25-11—S-tagged and Port-based service interface access classifications	993
Figure 25-12—I-tagged service interface access protection classifications	994
Figure 25-13—Internal organization of the MAC sublayer in a PBB-TE Region	996
Figure 25-14—PBB-TE Region	998
Figure 25-15—Transparent service interface	1000
Figure 25-16—Transparent service interface equipment	1000
Figure 26-1—PBBN example	1002
Figure 26-2—CFM shim model	1009

Figure 26-3—CFM example applied to a Port-based and S-tagged Service Interface.....	1010
Figure 26-4—CFM example applied to an I-tagged Service Interface.....	1011
Figure 26-5—CFM example applied to a hierachal E-NNI, CBP-PIP Demarc	1012
Figure 26-6—CFM example applied to a peer E-NNI, CBP-PIP	1013
Figure 26-7—Independent ESPs using the same ESP-DAs and ESP-VIDs	1017
Figure 26-8—PBB-TE MEP placement in a CBP.....	1017
Figure 26-9—Protection switching architecture.....	1020
Figure 26-10—PBB-TE point-to-point protection switching.....	1022
Figure 26-11—Mapping data traffic to the protection entity	1023
Figure 26-12—Relationships of the Protection switching state machines—overview	1024
Figure 26-13—Hold-off state machine.....	1028
Figure 26-14—Clear Manual Switch state machine	1028
Figure 26-15—Service Mapping state machine	1029
Figure 29-1—Forward path test (FPT)	1034
Figure 29-2—Return path test	1035
Figure 29-3—Combination of forward path test and return path test.....	1036
Figure 29-4—Detailed Functions of Reflection Responder	1037
Figure 29-5—RFM Receiver on an non-MP	1040
Figure 29-6—Return Path Decapsulator Responder	1041
Figure 29-7—RR Filter state machine.....	1046
Figure 29-8—RR Encapsulation state machine	1047
Figure 29-9—RR Transmit state machine	1047
Figure 29-10—RFM Receiver state machine	1048
Figure 29-11—Decapsulator Responder state machine.....	1051
Figure 30-1—Congestion detection in QCN CP	1057
Figure 30-2—Sampling (reflection) probability in QCN CP as a function of Fb	1058
Figure 30-3—QCN RP operation	1058
Figure 30-4—Byte Counter and Timer interaction with Rate Limiter	1060
Figure 30-5—CP-RP peering in VLAN Bridged Network.....	1063
Figure 30-6—CP-RP peering in Provider Backbone Bridged Network	1063
Figure 31-1—Congestion Points and congestion aware queues in a bridge.....	1065
Figure 31-2—Congestion aware queue functions in an end station	1066
Figure 31-3—Per-CNPV station function	1068
Figure 32-1—Congestion Notification Domain defense state machine	1081
Figure 32-2—RP rate control state machine.....	1093
Figure 32-3—CP-RP peering in any hierarchical Bridged Network	1094
Figure 34-1—Queuing model for a Talker station	1106
Figure 35-1—Operation of MSRP	1109
Figure 35-2—Format of the components of the reservation FirstValue fields	1119
Figure 35-3—Format of the components of the Domain FirstValue.....	1124
Figure C-1—CSN Backbone	1197
Figure C-2—Bridge's CSN Model for Bandwidth Reservation.....	1198
Figure C-3—Talker MSRPDU flow	1199
Figure C-4—Listener MSRPDU flow	1199
Figure C-5—IEEE DMN Device Attribute IE	1201
Figure C-6—DMN ConfirmationTransaction	1203
Figure C-7—Bandwidth Reservation - Bridge Model for IEEE 802.11 BSS (STA Downstream Port)..	1206
Figure C-8—Bandwidth Reservation - Bridge Model for IEEE 802.11 BSS (STA Upstream Port).....	1206
Figure C-9—Bandwidth Reservation - Bridge Model for IEEE 802.11 BSS (Direct Link Setup).....	1207
Figure C-10—MSRP/802.11 Query Flows	1207
Figure C-11—MSRP/802.11 Talker STA to Listener STA Reservation Flows.....	1208
Figure C-12—MSRP/802.11 “Bridged” Listener to Talker STA Reservation Flows.....	1209
Figure C-13—MSRP/802.11 Listener STA to “Bridged” Talker Reservation Flows.....	1209
Figure D-1—Port VLAN ID TLV Format	1215

Figure D-2—Port And Protocol VLAN ID TLV Format	1215
Figure D-3—VLAN Name TLV format.....	1216
Figure D-4—Protocol Identity TLV format	1217
Figure D-5—VID Usage Digest TLV format.....	1218
Figure D-6—Management VID TLV format	1218
Figure D-7—Link Aggregation TLV format.....	1219
Figure D-8—Congestion Notification TLV format.....	1219
Figure F-1—Connecting independent VLANs—1	1261
Figure F-2—Connecting independent VLANs—2	1262
Figure F-3—Duplicate MAC Addresses	1262
Figure F-4—Asymmetric VID use: “multi-netted server”	1263
Figure F-5—Asymmetric VLAN use: “Rooted-Multipoint”.....	1265
Figure G-1—Example of IEEE 802.3 MAC frame format	1269
Figure H-1—Static filtering inconsistency	1273
Figure H-2—Interoperability with IEEE 802.1D Bridges: example 1	1274
Figure H-3—Interoperability with IEEE 802.1D Bridges: example 2	1275
Figure H-4—Interoperability between Port-based and Port-and-Protocol-based classification.....	1277
Figure J-1—Up MPs in a CFM Port	1292
Figure K-1—TPMR as UNI demarcation device	1295
Figure K-2—TPMRs with aggregated links	1296
Figure K-3—Multiple TPMRs.....	1296
Figure K-4—Recovery at the end of a chain	1297
Figure K-5—Near simultaneous recoveries	1298
Figure K-6—Near simultaneous failure and recovery	1298
Figure K-7—Loss with quick recovery	1299
Figure L-1—Credit-based shaper operation—no conflicting traffic	1302
Figure L-2—Credit-based shaper operation—conflicting traffic	1303
Figure L-3—Credit-based shaper operation—burst traffic.....	1304
Figure L-4—Interference and latency.....	1308
Figure L-5—Burst behavior and credit.....	1308
Figure L-6—Fan-in scenario	1312
Figure L-7—Permanent delay scenario	1313
Figure L-8—Building up buffer occupancy—1	1313
Figure L-9—Building up buffer occupancy—2	1314
Figure L-10—Building up buffer occupancy—3	1314
Figure L-11—Building up buffer occupancy—4	1314

Tables

Table 6-1—Priority to MAC service class mapping	64
Table 6-2—MAC service class to priority mapping.....	65
Table 6-3—Priority Code Point encoding	71
Table 6-4—Priority Code Point decoding	71
Table 6-5—Priority regeneration.....	72
Table 6-6—Default SRP domain boundary port priority regeneration override values.....	73
Table 6-7—Service Access Priority	85
Table 6-8—Encapsulated Addresses EtherType	92
Table 8-1—C-VLAN component Reserved addresses	116
Table 8-2—S-VLAN component Reserved addresses	117
Table 8-3—TPMR component Reserved addresses	117
Table 8-4—Recommended priority to traffic class mappings.....	119
Table 8-5—Transmission selection algorithm identifiers	120
Table 8-6—Ageing time parameter value	127
Table 8-7—Combining Static and Dynamic Filtering Entries for an individual MAC Address	133
Table 8-8—Combining Static Filtering Entry and MAC Address Registration Entry for “All Group Addresses” and “All Unregistered Group Addresses”	134
Table 8-9—Forwarding or Filtering for specific group MAC Addresses	135
Table 8-11—Determination of whether a Port is in a VID’s member set	136
Table 8-10—Forwarding or Filtering with Dynamic Reservation Entries	136
Table 8-12—Standard LLC address assignment	140
Table 8-13—Continuity Check Message Group Destination MAC Addresses	147
Table 8-14—Linktrace Message Group Destination MAC Addresses.....	147
Table 9-1—IEEE 802.1Q EtherType allocations	150
Table 9-2—Reserved VID values	150
Table 9-3—Reserved I-SID values.....	152
Table 10-1—MRP application addresses	160
Table 10-2—MRP EtherType values	161
Table 10-3—Applicant state table	174
Table 10-4—Registrar state table	175
Table 10-5—LeaveAll state table	176
Table 10-6—PeriodicTransmission state table	176
Table 10-7—MRP timer parameter values	177
Table 12-1—Bandwidth Availability Parameter Table row elements.....	310
Table 12-2—Transmission Selection Algorithm Table row elements	310
Table 12-3—Priority Regeneration Override Table row elements.....	311
Table 12-4—CN component managed object row elements	312
Table 12-5—CN component priority managed object row elements	312
Table 12-6—CN Port priority managed object row elements	313
Table 12-7—Congestion Point managed object row elements.....	314
Table 12-9—Reaction Point group managed object row elements	315
Table 12-8—Reaction Point port priority managed object row elements	315
Table 12-10—SRP Bridge Base Table row elements.....	316
Table 12-11—SRP Bridge Port Table row elements.....	316
Table 12-13—SRP Stream Table row elements	317
Table 12-12—SRP Latency Parameter Table row elements	317
Table 12-14—SRP Reservations Table row elements.....	318
Table 13-1—Configuration Digest Signature Key	333
Table 13-2—Sample Configuration Digest Signature Keys.....	334
Table 13-3—Bridge and Port Priority values	348
Table 13-4—Port Path Cost values.....	348

Table 13-5—Timer and related parameter values	356
Table 17-1—Structure of the MIB modules.....	410
Table 17-2—IEEE8021-TC MIB Structure	411
Table 17-3—IEEE8021-BRIDGE MIB structure and relationship to IETF RFC 4188 and this standard	413
Table 17-4—IEEE 802.1D objects not in the IEEE8021-BRIDGE MIB	417
Table 17-5—IEEE8021-SPANNING-TREE MIB structure and relationship to IETF RFC 4318 and this standard	417
Table 17-6—Clause 12 objects not in the IEEE8021-SPANNING-TREE MIB	419
Table 17-7—IEEE8021-QBRIDGE MIB structure and relationship to IETF RFC 4363 and this standard	420
Table 17-8—Clause 12 management not in IEEE8021-Q-BRIDGE MIB	425
Table 17-9—IEEE8021-PB MIB structure and relationship to this standard	426
Table 17-10—IEEE8021-MSTP MIB structure and relationship to this standard.....	428
Table 17-11—IEEE8021-CFM MIB correspondence between variables, managed objects, and MIB objects.....	431
Table 17-12—IEEE8021-CFM-V2 correspondence between variables, managed objects, and MIB objects.....	435
Table 17-13—IEEE8021-PBB MIB structure and relationship to this standard	437
Table 17-14—IEEE8021-DDCFM MIB structure and relationship to this standard	440
Table 17-15—IEEE8021-PBBTE MIB Structure and relationship to this standard	442
Table 17-16—Example of ieee8021PbbTeTeSidTable	444
Table 17-17—IEEE8021-TPMR MIB Structure and relationship to this standard	445
Table 17-18—FQTSS MIB structure and object cross reference.....	447
Table 17-19—Variables, managed object tables, and MIB objects.....	448
Table 17-20—SRP MIB structure and object cross reference.....	450
Table 17-21—PBB-TE required MIB compliances	460
Table 17-22—Sensitive managed objects: tables and notifications	466
Table 17-23—Sensitive managed objects: variables in dot1agCfmMdTable	466
Table 17-24—Sensitive managed objects (of DDCFm): tables and notifications	468
Table 17-25—Sensitive managed objects (of DDCFm) for read.....	468
Table 17-26—Provider Bridge service interface parameters	481
Table 17-27—Provider Backbone Bridge service interface parameters	485
Table 19-1—Actions taken by MP OpCode Demultiplexers	857
Table 19-2—SAP use for Linktrace Messages and Linktrace Replies.....	864
Table 20-1—Fault Alarm defects and priorities	869
Table 20-2—Deriving enableRmepDefect and Port Status TLV in a Bridge	879
Table 21-1—CFM PDU Encapsulation: Length/Type Media	920
Table 21-2—CFM PDU Encapsulation: LLC Media	920
Table 21-3—Common CFM Header format	921
Table 21-5—TLV format.....	922
Table 21-4—OpCode Field range assignments	922
Table 21-7—Organization-Specific TLV format	923
Table 21-6—Type Field values	923
Table 21-8—Sender ID TLV format	924
Table 21-11—Interface Status TLV format.....	926
Table 21-12— Interface Status TLV values	926
Table 21-9—Port Status TLV format	926
Table 21-10—Port Status TLV values.....	926
Table 21-13—Data TLV format	927
Table 21-14—End TLV format	927
Table 21-15—Continuity Check Message format	928
Table 21-16—CCM Interval field encoding.....	929

Table 21-19—Maintenance Domain Name Format	930
Table 21-17—CCM MAID field format: Maintenance Domain present	930
Table 21-18—CCM MAID field format: Maintenance Domain not present	930
Table 21-20—Short MA Name Format.....	931
Table 21-21—Loopback Message and Loopback Reply formats.....	932
Table 21-22—PBB-TE MIP TLV format.....	933
Table 21-23—Linktrace Message format	934
Table 21-24—Linktrace Message Flags field.....	935
Table 21-25—LTM Egress Identifier TLV format.....	936
Table 21-26—Linktrace Reply format	936
Table 21-27—Linktrace Reply Flags field	937
Table 21-28—Relay Action field values	937
Table 21-29—LTR Egress Identifier TLV format	938
Table 21-30—Reply Ingress TLV format	939
Table 21-31—Ingress Action field values	939
Table 21-32—Reply Egress TLV format	940
Table 21-33—Egress Action field values.....	941
Table 22-1—MEP creation	954
Table 22-2—MIP creation	955
Table 22-3—Bandwidth required for CCMs for 1 MA	958
Table 22-4—Bandwidth required for CCMs for 1000 MAs	959
Table 23-1—Time sequence diagram symbols	969
Table 23-2—MSP performance parameters	974
Table 23-3—MSP EtherType assignment	980
Table 23-4—MSP Packet Types.....	981
Table 26-1—Backbone Service Instance Group address OUI	1005
Table 26-8—Protection Requests Hierarchy	1025
Table 29-1—RFM format	1052
Table 29-2—SFM format	1053
Table 32-1—LLDP instance selection managed object overrides.....	1074
Table 32-2—CND defense mode selection managed object overrides	1074
Table 32-3—Determining cnpdIsAdminDefMode and cnpdDefenseMode.....	1080
Table 32-4—Correspondence of QCN protocol and CCF message fields	1082
Table 32-5—NewCpSampleBase() return value as a function of cpFb	1085
Table 33-3—Congestion Notification Message Encapsulation: Length/Type Media.....	1097
Table 33-1—Congestion Notification Tag Encapsulation: Length/Type Media.....	1097
Table 33-2—Congestion Notification Tag Encapsulation: LLC Media.....	1097
Table 33-5—Congestion Notification Message PDU	1098
Table 33-4—Congestion Notification Message Encapsulation: LLC Media	1098
Table 34-1—Recommended priority to traffic class mappings for SR classes A and B	1105
Table 34-2—Recommended priority to traffic class mappings for SR class B only.....	1105
Table 35-1—AttributeType Values	1115
Table 35-2—AttributeLength Values	1115
Table 35-3—FourPackedEvent Values	1116
Table 35-4—MSRP FirstValue NumberOfValues example	1117
Table 35-5—TSpec Components Examples	1121
Table 35-6—Reservation Failure Codes.....	1123
Table 35-7—SR class ID	1124
Table 35-8—Summary of Talker primitives	1126
Table 35-9—Summary of Listener primitives.....	1126
Table 35-11—Incoming Listener Attribute Propagation per port	1131
Table 35-10—Talker Attribute Propagation per port	1131
Table 35-12—Updating Dynamic Reservation Entries	1132
Table 35-13—Updating <i>operIdleSlope(N)</i>	1132

Table 35-14—Listener Declaration Type Summation.....	1133
Table C-1—SRP to MoCA PQoS Transaction mapping.....	1204
Table C-2—SRP TSpec to MoCA TSPEC mapping.....	1205
Table C-3—SRP StreamID to MoCA PQoS Flow transaction mapping	1205
Table C-4—SRP to MLME QoS Services mapping	1211
Table C-5—EDCA-AC for AV Streams	1212
Table C-6—HCCA for AV Streams.....	1213
Table D-1— IEEE 802.1 Organizationally Specific TLVs	1214
Table D-2—Port and protocol capability/status	1215
Table D-3—Link aggregation capability/status.....	1219
Table D-4—IEEE 802.1 extension MIB object group conformance requirements.....	1222
Table D-5—IEEE 802.1/LLDP extension MIB object cross reference.....	1222
Table E-6—State machine symbols.....	1259
Table I-1—Traffic type to traffic class mapping	1281
Table I-2—Traffic type acronyms	1282
Table I-3—Defining traffic types	1283
Table I-4—Defining traffic types—Credit-based shaper support of one SR class.....	1284
Table I-5—Defining traffic types—Credit-based shaper support of two SR classes	1284
Table I-6—Priority Code Point encoding	1286
Table I-7—Priority Code Point decoding.....	1286
Table J-1—Provider MD Level allocation	1288
Table J-2—IEEE / ITU-T terminology differences.....	1288

IEEE Standard for Local and metropolitan area networks—

Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks

IMPORTANT NOTICE: This standard is not intended to ensure safety, security, health, or environmental protection. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

IEEE 802® Local Area Networks (LANs)¹ of all types can be connected together with Media Access Control (MAC) Bridges, as specified in IEEE Std 802.1D™.^{2,3} This standard defines the operation of Bridges that permit the definition, operation, and administration of Virtual LANs (VLANs) within Virtual Bridged Local Area Networks.

This standard further extends the specification of VLAN-aware MAC Bridges to enable a service provider organization to use a common infrastructure of Bridges and LANs to offer the equivalent of separate LANs, Bridged, or Virtual Bridged Local Area Networks to independent customer organizations.

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.

¹IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

²Information on references can be found in Clause 2.

³Throughout this standard, references to IEEE Std 802.1D, without qualification as to the date of publication, refer to the most recent revision or edition of the standard that is identified in the references section (Clause 2). In those cases where the reference is intended to be to an earlier published version of the standard, the reference is qualified by the publication date of that revision or edition.

The data-driven and data-dependent connectivity fault management capabilities include tools to facilitate diagnosis and isolation of faults sensitive to, or caused by, particular data patterns in transmitted frames.

This standard specifies protocols, procedures, and managed objects that support congestion management of long-lived data flows within network domains with a bandwidth-delay product on the order of 5 Mbits or less. Such flows are typically encountered in data centers, backplane fabrics, computing clusters, and storage networks. This is achieved by enabling bridges to signal congestion to end stations capable of transmission rate limiting to avoid frame loss. This mechanism enables support for higher layer protocols that are highly loss or latency sensitive. VLAN tag encoded priority values are allocated to segregate frames subject to congestion control, allowing simultaneous support of both congestion controlled and other higher layer protocols. This standard does not specify communication or reception of congestion notification information to or from end stations outside the congestion controlled domain or encapsulation of frames from those end stations across the domain.

This standard specifies protocols, procedures and managed objects, usable by existing higher layer mechanisms, that allow network resources to be reserved for specific traffic streams traversing a bridged local area network. It characterizes resource requirements of traffic streams to a level sufficient for bridges to determine the required resources and provides a mechanism for dynamic maintenance of those resources.

1.1 Scope

This standard specifies Media Access Control (MAC) Bridges that interconnect individual Local Area Networks (LANs), each supporting the IEEE 802 MAC service using a different or identical media access control method, to provide Bridged Local Area Networks and Virtual LANs (VLANs).

1.2 Purpose

MAC Bridges, as specified by this standard, allow the compatible interconnection of information technology equipment attached to separate individual LANs.

1.3 Introduction

For the purpose of compatible interconnection of information technology equipment using the IEEE 802 MAC Service supported by interconnected IEEE 802 standard LANs using different or identical media access control methods, this standard specifies the operation of MAC Bridges that support Virtual LANs (VLANs). To this end it

- a) Positions the support of VLANs within an architectural description of the MAC Sublayer;
- b) Defines the principles of operation of the VLAN-aware Bridge in terms of the support and preservation of the MAC Service, and the maintenance of Quality of Service;
- c) Specifies an Enhanced Internal Sublayer Service provided to the Media Access Independent functions that provide frame relay in a VLAN-aware Bridge;
- d) Establishes the principles and a model of Virtual Bridged Local Area Network operation;
- e) Identifies the functions to be performed by VLAN-aware Bridges, and provides an architectural model of the operation of a Bridge in terms of Processes and Entities that provide those functions;
- f) Specifies a frame format that allows a VLAN Identifier (VID) and priority information to be carried by VLAN tagged user data frames;
- g) Specifies the rules that govern the addition or removal of VLAN tags to and from user data frames;
- h) Specifies the rules that govern the ability to carry user data in either Canonical format or Noncanonical format in VLAN-tagged frames;

NOTE 1—The meanings of the terms *Canonical format* and *Noncanonical format* are discussed in IEEE Std 802.⁴

- i) Establishes the requirements for automatic configuration of VLAN topology;
- j) Establishes the requirements for VLAN-aware Bridge Management in a Virtual Bridged Local Area Network, identifying managed objects and defining management operations;
- k) Defines SMIv2 (IETF STD 58) MIB modules for the management of VLAN-aware Bridge capabilities including Spanning Tree Protocols and Provider Bridges.
- l) Defines the operation of the Multiple Spanning Tree algorithm and protocol (MSTP);
- m) Describes the protocols and procedures necessary to support interoperation between MST and SST Bridges in the same Virtual Bridged Local Area Networks;
- n) Specifies the requirements to be satisfied by equipment claiming conformance to this standard.

To enable a service provider to use a Virtual Bridged Local Area Network to provide separate instances of the IEEE 802® MAC Service, MAC Internal, and Enhanced Internal Sublayer Services to multiple independent customers, in a manner that does not require cooperation among the customers and that requires a minimum of cooperation between the customers and the provider of the MAC Service, this standard further specifies the operation of Provider Bridges. To this end it

- o) Differentiates Customer VLANs (C-VLANs) that are under the administrative control of a single customer of a service provider, from the Service VLANs (S-VLANs) that are used by a service provider to support different customers.
- p) Specifies VLAN TAG formats for both C-VLANs and S-VLANs, allowing each to be distinguished and separately applied and administered by customers and by a service provider.
- q) Specifies the functionality of a generic VLAN-aware bridge component within a system and the specific requirements of derived C-VLAN and S-VLAN components.
- r) Specifies a VLAN Bridge as comprising a single C-VLAN component, and a Provider Bridge as encompassing bridges that comprise a single S-VLAN component and no C-VLAN components (S-VLAN Bridge) or a single S-VLAN component and one or more C-VLAN components (Provider Edge Bridge).
- s) Specifies parameters and mappings that allow the Enhanced Internal Sublayer Service to support traffic classes that comprise distinct aggregate flows supporting different QoS characteristics and provide independent guarantees to different customers, through support of priority and drop precedence marking.
- t) Specifies the incorporation of flow metering, transmission queue management, and transmission selection algorithms within the forwarding process of a Bridge.
- u) Positions the support of S-VLANs within the architectural description of the MAC Sublayer and specifies their relationship to media access method dependent functions and to the media independent functions used by customers to administer their networks, including the support of C-VLANs.
- v) Allocates the reserved multicast addresses to media access method dependent, provider network, and customer network functions, specifying the filtering to be applied in each type of VLAN-aware bridge component.
- w) Defines the principles of network operation in terms of the support and preservation of the MAC Service, and the maintenance of Quality of Service for each service instance, including the segregation of data belonging to different organizations.
- x) Specifies customer interfaces to a Provider Bridged Network in terms of the operation and configuration of the VLAN-aware bridge components of Provider Bridges, including interfaces that
 - 1) Provide access to a single service instance through a Bridge Port
 - 2) Allow a customer to select amongst and identify service instances by Customer VLAN Identifier (C-VID)
 - 3) Allow a customer to select amongst and identify service instances by Service VLAN Identifier (S-VID)
 - 4) Support customer signaling of priority information on a frame by frame basis

⁴Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

- 5) Multiplex service instances over LANs that provide access to a provider network
- 6) Support fault tolerance through redundant provision of access LANs and equipment.
- y) Describes the functions to be performed within the Provider Bridged Network in order to support and maintain the connectivity provided to customer service instances.
- z) Establishes the requirements for Bridge Management in the Provider Bridged Network, identifying the managed objects and defining the management operations.
- aa) Specifies performance requirements, and recommends default values and applicable ranges for the operational parameters of a Provider Bridge.

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:

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

This standard specifies protocols, procedures, and managed objects to allow support of provisioning systems that explicitly select traffic engineered paths within Provider Backbone Bridged Networks by allowing a network operator to disable unknown destination address forwarding, source address learning and spanning tree protocols for administratively selected VIDs, while allowing other network control protocols to dynamically determine active topologies for other services. These interoperable capabilities are supported by management of individual bridges by SNMP using an SMIPv2 MIB, by extensions to the other control protocols specified in this standard, by the use of CFM with the addresses and VIDs that specify traffic engineered connections, and by 1:1 path protection switching capable of load sharing. To this end, it:

- ad) Enables construction of active topologies by an external agent that is responsible for setting up Ethernet Switched Paths (ESPs) by splitting the B-VID space between distributed spanning tree protocols and provisioned control.
- ae) Supports discard of frames with unknown destination addresses for B-VIDs under provisioned control.
- af) Supports the operation of Continuity Check, Loopback, and Linktrace protocols on provisioned traffic engineered paths.
- ag) Supports 1:1 protection switching capable of load sharing for Traffic Engineered service instances.
- ah) Provides required extension to SNMP management by SMIPv2 MIB modules.

This standard does not specify operation of Ethernet Switched Paths (ESPs) through multiple Provider Backbone Bridge Traffic Engineered (PBB-TE) Regions. All the Backbone Edge Bridges specified for use in a PBB-TE Region are IB-BEBs.

In addition, this standard specifies protocols, procedures, and managed objects to support the Multiple Registration Protocol (MRP). MRP allows participants in an MRP Application to register attributes with other participants in a Bridged Local Area Network. Two applications are defined—one to register VIDs [Multiple VLAN Registration Protocol (MVRP)] and one to register MAC addresses [Multiple MAC Registration Protocol (MMRP)]. MVRP will furthermore provide for the rapid healing of network failures without interrupting services to unaffected VLANs. To this end, it specifies the following:

- ai) MRP and the operation of MRP entities.⁵
- aj) The generic frame formats used in MRP exchanges.

⁵MRP replaces the Generic Attribute Registration Protocol (GARP), defined in IEEE Std 802.1D, that was used to support GVRP and GMRP in earlier revisions of IEEE Std 802.1Q. Similarly, GVRP and GMRP are replaced by MVRP and MMRP, respectively.

- ak) The Multiple MAC Registration Protocol (MMRP) application of MRP, and the frame formats that it uses.
- al) The Multiple VLAN Registration Protocol (MVRP) application of MRP, and the frame formats that it uses.

To allow scaling of Provider Networks to at least 2^{24} Service Virtual LANs, this standard further specifies the operation of Provider Backbone Bridges (PBBs) by means of an architecture and bridge protocols compatible and interoperable with Provider Bridged Network protocols and equipment, allowing interconnection of multiple Provider Bridged Networks. To this end, it

- am) Introduces Backbone Edge Bridges that, by exchanging backbone frames that encapsulate the addresses, VLAN tags, and data of customer frames, support the virtual, media independent, equivalent of a number of independent instances of the service provided by media dependent frame transmission procedures.
- an) Extends the parameters of the Internal Sublayer Service (ISS) and Enhanced Internal Sublayer Service (EISS) to include a connection identifier, capable of referencing the backbone addresses and other parameters, used to convey customer frames from one Backbone Edge Bridge (BEB) to all, or one of, the other BEBs supporting a particular backbone service instance.
- ao) Specifies the format of the Backbone Service Instance tag (I-TAG) that encapsulates the customer addresses, and introduces a Backbone Service Instance Identifier (I-SID) that allows each BEB to support a number of backbone service instances and permits the unambiguous identification of up to 2^{24} backbone service instances within a single Provider Backbone Bridged Network (PBBN).
- ap) Provides a model of Backbone Edge Bridge operation in terms of VLAN-aware bridge components that allows the use of Provider Bridges as Backbone Core Bridges, with PBBN traffic carried as frames containing I-TAGs on particular Backbone VLANs (B-VLANs) potentially coexisting with PBN traffic carried as frames without I-TAGs on other Backbone VLANs.
- aq) Specifies the interfaces that a Provider Backbone Bridged Network can provide to transport service frames. These comprise a Port-based service interface that assigns all received untagged and priority-tagged frames to a single S-VLAN transported over a single backbone service instance, an S-tagged service interface capable of mapping individual S-VLANs to different backbone service instances, and an I-tagged service interface capable of mapping frames from one set of backbone service instances to another.
- ar) Describes the use of redundant bridges and access LANs to protect backbone service access against failure of any of those systems or components.
- as) Specifies the management of Backbone Edge Bridges in terms of the model of operation [item ap] above], making use of defined management objects for the individual VLAN-aware bridge components, and adding managed objects to facilitate service creation.
- at) Describes the use of connectivity fault management (CFM) to detect and isolate faults in the connectivity provided to individual S-VLANs across the PBBN, in the connectivity provided to the group of S-VLANs supported by a single backbone service instance (identified by an I-SID), and in the connectivity provided to individual B-VLANs within the backbone itself.
- au) Specifies extensions to the Multiple Spanning Tree Protocol (MSTP) to allow network administrators to protect against loops through peered PBBNs without requiring coupling of spanning trees that operate independently for each PBBN.

This standard specifies connectivity fault management protocols, procedures, and managed objects that provide confirmation of successful transmission of frames conveying specified data. This capability supports diagnosis of faults sensitive to, or caused by, particular data patterns, and their isolation to part of the transmission path. Connectivity verification can be carried out from any single point with bridged connectivity to maintenance points on the path, can isolate failures to communicate in a specific direction, and can be carried out while service is being provided to other users of the data path. To this end it

- av) Defines the extensions to connectivity fault management capabilities defined by Clause 18 through Clause 22 to facilitate diagnosis and isolation of faults sensitive to, or caused by, particular data patterns in frames transmitted by a service user;
- aw) Describes the protocols and procedures for data-driven and data-dependent connectivity fault management (DDCFM).

Additionally, this standard specifies the function of a Two-Port MAC Relay (TPMR), along with protocols and procedures that support its operation. A TPMR is a type of bridge that has only two externally accessible Bridge Ports, and supports a subset of the functionality of a MAC Bridge. A TPMR is transparent to all frame-based media independent protocols, except those explicitly addressed to it and those that are destined for reserved MAC addresses that the relay function of the TPMR is defined not to forward. It is remotely manageable through at least one of its external MACs, and signals a failure of either MAC's LAN through the other MAC. A TPMR should only be attached to point-to-point LANs. The conformance requirements for a TPMR are stated in 5.13 and 5.14.

This standard allows bridges to provide performance guarantees for time-sensitive (i.e., bounded latency and latency variation) loss-sensitive real-time audio video (AV) data stream transmission (AV traffic). It specifies priority regeneration and controlled bandwidth queue draining algorithms. VLAN tag encoded priority values are allocated, in aggregate, to segregate frames among queues that support AV traffic and queues that support non-AV traffic, allowing simultaneous support of both AV traffic and other bridged traffic over and between wired and wireless Local Area Networks (LANs). To this end, it

- ax) Defines status parameters that allow the boundaries of a Stream Reservation Protocol (SRP—see Clause 35) domain (6.6.4) to be identified and maintained
- ay) Specifies how the priority information in frames received at SRP domain boundary ports is regenerated

NOTE 2—The priorities in frames transmitted from outside an SRP domain to a Bridge inside an SRP domain are remapped in order to ensure that traffic that is not associated with a reservation does not disrupt traffic that is associated with a reservation. Hence, traffic entering an SRP domain that uses priority code point values associated with reserved traffic classes will be remapped to priority code point values that are not associated with reserved traffic classes.

- az) Specifies how priority information is used to determine the traffic classes to be used for time-sensitive streams
- ba) Defines a credit-based shaper algorithm to shape traffic in accordance with stream reservations

NOTE 3—The credit-based shaper algorithm operates on the outbound queues; the mechanisms specified for the support of time-sensitive AV traffic do not involve any form of ingress metering or policing.

This standard specifies protocols, procedures, and managed objects to support congestion notification. These allow a Virtual Bridged Local Area Network or a portion thereof, with a limited bandwidth-delay product, to transfer long-lived data flows with a significantly reduced chance of frame loss compared to a network without congestion notification. To this end, it

- bb) Defines a means for VLAN-aware Bridges that support congestion notification to form Congestion Managed Domains within a Virtual Bridged LAN.
- bc) Defines a means for detecting congested queues in end stations and VLAN-aware Bridges, for signaling such congestion to the end stations sourcing the frames causing the congestion, and for those end stations to control the rate of transmission of those frames.

To enable the end-to-end management of resource reservation for QoS guaranteed streams, this standard further specifies protocols, procedures and managed objects, usable by existing higher layer mechanisms, that allow network resources to be reserved for specific traffic streams traversing a bridged local area network. To this end it:

- bd) Specifies the use of Dynamic Reservation Entries (8.8.7) in the filtering database to control the forwarding of frames associated with a particular Stream.
- be) Specifies a Stream Reservation Protocol (SRP). SRP facilitates the registration, de-registration and maintenance of stream reservation information in relevant bridges to establish end-to-end stream paths.

1.4 VLAN aims and benefits

VLANs aim to offer the following benefits:

- a) VLANs facilitate easy administration of logical groups of stations that can communicate as if they were on the same LAN. They also facilitate easier administration of moves, adds, and changes in members of these groups.
- b) Traffic between VLANs is restricted. Bridges forward unicast, multicast, and broadcast traffic only on individual LANs that serve the VLAN to which the traffic belongs.
- c) As far as possible, VLANs maintain compatibility with existing bridges and end stations.
- d) If all Bridge Ports are configured to transmit and receive untagged frames (3.194), bridges will work in plug-and-play IEEE 802.1D mode. End stations will be able to communicate throughout the network.

NOTE—Whether a Bridge will operate in IEEE 802.1D mode depends on the configuration of the various Bridge Port parameters (8.4) and the Filtering Database (8.8). A Bridge in its default configuration is transparent to untagged frames (3.194) but is not transparent to tagged frames (3.182), so the operation of such Bridges in the presence of tagged traffic differs from that of an IEEE 802.1D Bridge. If the configuration settings of Bridges are changed from the default values defined in this standard, then transparency with respect to untagged frames may also be affected.

2. Normative references

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.

ANSI X3.159, American National Standards for Information Systems—Programming Language—C.⁶

IEEE Std 802[®], IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture.^{7, 8}

IEEE Std 802.1ABTM-2009, IEEE Standard for Local and metropolitan area networks—Station and Media Access Control Connectivity Discovery.

IEEE Std 802.1DTM, 1998 Edition [ISO/IEC 15802-3:1998], IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 3: Media Access Control (MAC) Bridges.

IEEE Std 802.1DTM, IEEE Standard for Local and metropolitan area networks—Media Access Control (MAC) Bridges.

IEEE Std 802.1XTM, IEEE Standards for Local and Metropolitan Area Networks—Port Based Network Access Control.

IEEE Std 802.1AXTM-2008, IEEE Standard for Local and metropolitan area networks—Link Aggregation.

IEEE Std 802.3TM, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications.

IEEE Std 802.11TM, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

IEEE Std 802.17TM, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 17: Resilient packet ring (RPR) access method and physical layer specifications.

IEEE Std 802.20TM, IEEE Std for Local and Metropolitan Area Networks. Part 20: Air Interface for Mobile Broadband Wireless Access Systems Supporting Vehicular Mobility - Physical and Media Access Control Layer Specification.

IETF RFC 1035 (STD 13), Domain Names: Implementation and Specification.⁹

IETF RFC 1042 (Feb. 1988), A Standard for the Transmission of IP Datagrams over IEEE 802 Networks, Postel, J., and Reynolds, J.¹⁰

⁶ANSI publications are available from the Sales Department, American National Standards Institute, 25 West 43nd Street, 4th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

⁷IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854-4141, USA (<http://standards.ieee.org/>).

⁸The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

⁹IETF documents are available at <http://www.ietf.org/>.

¹⁰Internet RFCs are retrievable by FTP at ds.internic.net/rfc/rfcnnnn.txt (where nnnn is a standard's publication number such as 1042), or call InterNIC at 1-800-444-4345 for information about receiving copies through the mail.

IETF RFC 2104 (Feb. 1997), HMAC: Keyed-Hashing for Message Authentication, Krawczyk, H., Bellare, M., and Canetti, R.¹¹

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

IETF RFC 2205, Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification.

IETF RFC 2578 (STD 58), Structure of Management Information Version 2 (SMIV2), McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, April 1999.

IETF RFC 2579 (STD 58), Textual Conventions for SMIV2, McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, April 1999.

IETF RFC 2580 (STD 58), Conformance Statements for SMIV2, McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and S. Waldbusser, April 1999.

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

IETF RFC 2750, RSVP Extensions for Policy Control.

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

IETF RFC 3410, Introduction and Applicability Statements for Internet-Standard Management Framework, Case, J., Mundy, R., Partain, D. and B. Stewart, December 2002.

IETF RFC 3411 (December 2002), *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*, Harrington, D., Presuhn, R., and Wijnen, B.

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.

¹¹IETF documents are available at <http://www.ietf.org>.

IETF RFC 4789, Simple Network Management Protocol (SNMP) over IEEE 802 Networks, J. Schoenwaelder, T. Jeffree, November 2006.

ISO 6937-2, Information technology—Coded graphic character set for text communication—Latin alphabet.¹²

ISO/IEC 7498-1, Information processing systems—Open Systems Interconnection—Basic Reference Model—Part 1: The Basic Model.

ISO/IEC 8802-2, Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 2: Logical link control.

ISO/IEC 8802-11, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications.

ISO/IEC 15802-1, Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Common specifications—Part 1: Medium Access Control (MAC) service definition.

ITU-T Recommendation X.690 (2002), Information Technology—ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER).¹³

ITU-T Recommendation Y.1731 (02/2008), OAM Functions and Mechanisms for Ethernet-based Networks.

Metro Ethernet Forum Technical Specification MEF 4, Metro Ethernet Network Architecture Framework—Part 1: Generic Framework, May 2004.¹⁴

Metro Ethernet Forum (MEF) Technical Specification MEF 10.2, Ethernet Service Attributes Phase 2, May 2009.

¹²ISO and ISO/IEC documents are available from the ISO Central Secretariat, 1 rue de Varembé, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse; and from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA.

¹³ITU-T publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (<http://www.itu.int/>).

¹⁴MEF publications are available at <http://www.metroethernetforum.org>.

3. Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary: Glossary of Terms & Definitions* should be consulted for terms not defined in this clause.¹⁵

This standard makes use of the following terms defined in IEEE Std 802:

- end station
- station

The following terms are specific to this standard or to this standard and IEEE Std 802.1D:

- 3.1 Active SAP:** The SAP, bounding an MP, that is on the side of the MP toward the monitored MA.
- 3.2 audio video (AV) traffic:** Traffic associated with audio and/or video applications that is sensitive to transmission latency and latency variation, and, for some applications, packet loss.
- 3.3 B-component (B-comp):** An S-VLAN component with one or more Customer Backbone Ports (CBPs).
- 3.4 Backbone Core Bridge (BCB):** An S-VLAN Bridge used within the core of a Provider Backbone Bridged Network.
- 3.5 Backbone Edge Bridge (BEB):** A system that encapsulates customer frames for transmission across a Provider Backbone Bridged Network.
- 3.6 backbone MAC address (B-MAC):** An individual MAC address associated with a Provider Instance Port and used in creating the MAC header of I-tagged frames transmitted across a Provider Backbone Bridged Network.
- 3.7 Backbone MAC Frame:** A LAN frame addressed with backbone MAC addresses.
- 3.8 backbone service instance:** An instance of the MAC service in a Provider Backbone Bridged Network, provided between two or more Virtual Instance Ports in Backbone Edge Bridges.
- 3.9 Backbone Service Instance Drop Eligibility Indicator (I-DEI):** A field of the Backbone Service Instance tag that indicates the drop eligibility of a frame in a backbone service instance.
- 3.10 Backbone Service Instance Identifier (I-SID):** A field of the Backbone Service Instance tag that identifies the backbone service instance of a frame.
- 3.11 Backbone Service Instance priority code point (I-PCP):** A field of the Backbone Service Instance tag that indicates the priority of a frame in a backbone service instance.
- 3.12 Backbone Service Instance tag (I-TAG):** A tag with an EtherType value allocated for “IEEE 802.1Q Backbone Service Instance tag EtherType.”
- 3.13 Backbone VLAN (B-VLAN):** A VLAN identified by a Backbone VLAN ID.
- 3.14 Backbone VLAN drop eligible indicator (B-DEI):** A field of a B-TAG that identifies the drop eligibility of the frame.

¹⁵The *IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at <http://shop.ieee.org/>.

3.15 Backbone VLAN ID (B-VID): A VLAN identifier conveyed in a B-TAG.

3.16 Backbone VLAN priority code point (B-PCP): A field of a B-TAG that indicates the Backbone VLAN priority of the frame.

3.17 Backbone VLAN tag (B-TAG): An S-TAG used in conjunction with backbone MAC addresses.

3.18 Backbone VLAN tagged frames: Frames that contain a B-TAG immediately following the Source MAC address field.

3.19 bidirectional protection switching: A protection switching architecture in which, for a failure, including a unidirectional failure (i.e., a failure affecting only one direction of transmission), both directions including the affected direction and the unaffected direction, are switched.

3.20 Boundary Port: A Bridge Port attaching an MST Bridge to a LAN that is not in the same region.

3.21 Bridge: A Bridge implemented in accordance with Clause 5 of this standard.

NOTE—This term defines a Bridge as specified in this standard. Where there is a need to refer generically to a bridge, being either a MAC Bridge as specified in IEEE Std 802.1D or a bridge as specified in this standard, the term is used without capitalization to indicate that the term is being used in the generic sense.

3.22 Bridge Port: A service access point (SAP) that provides the EISS to the MAC Relay Entity of a VLAN-aware Bridge component, or that provides the ISS to the MAC Relay Entity of a MAC Bridge, and the interface stack that supports that SAP (see 6.1.4, 8.5).

NOTE—In simple cases specific MAC procedures (6.7) provide the ISS (directly) or the EISS (using the procedures specified in 6.9), and each Bridge Port corresponds to a single and distinct LAN attachment. More complex interface stacks can aggregate multiple LANs to support a single Bridge Port, or can multiplex many Bridge Ports over a single LAN or set of LANs.

3.23 Bridged Local Area Network: A concatenation of individual IEEE 802 LANs interconnected by MAC Bridges.

NOTE—Unless explicitly specified, the use of the word “network” and the term “bridged network” in this standard refers to a Virtual Bridged Local Area Network or a Bridged Local Area Network. The terms “Virtual Bridged Local Area Network” and “Bridged Local Area Network” are not otherwise abbreviated. The term “Local Area Network” and the abbreviation LAN are used exclusively to refer to an individual LAN specified by a MAC technology without the inclusion of Bridges. This precise use of terminology within this specification allows a Bridged Local Area Network to be distinguished from an individual LAN that has been bridged to other LANs in the network. In more general usage, such precise terminology is not required, as it is an explicit goal of this standard that bridges are transparent to the users of the MAC Service.

3.24 C-tagged service interface: The interface provided by a Provider Edge Bridge to allow the attached customer to select between and identify service instances using the C-VID associated with transmitted and received frames.

3.25 C-VLAN Bridge: A VLAN Bridge.

3.26 C-VLAN component: A VLAN-aware bridge component with each Port supported by an instance of the EISS that can recognize, insert, and remove Customer VLAN tags.

3.27 Common and Internal Spanning Tree (CIST): The single Spanning Tree calculated by STP and RSTP and the logical continuation of that connectivity through MST Bridges and Regions, calculated by MSTP to ensure that all LANs in the Bridged Local Area Network are simply and fully connected.

3.28 Common Spanning Tree (CST): The single Spanning Tree calculated by STP, RSTP, and MSTP to connect MST Regions.

3.29 congestion-aware system: A bridge component or end station conforming to the congestion notification provisions of this standard.

3.30 Congestion Controlled Flow (CCF): A sequence of frames with the same priority, that the transmitting end station treats as belonging to a single flow, using a single Reaction Point to control the transmission rate of all those frames.

3.31 Congestion Notification Domain (CND): A connected subset of the end stations and VLAN-aware Bridges in a Virtual Bridged Network that are compatibly configured to support a Congestion Notification Priority Value. Multiple CNDs serving different Congestion Notification Priority Values can co-exist in a Virtual Bridged Network; the bridges, end stations and Ports included in the CNDs do not necessarily coincide.

3.32 Congestion Notification Message (CNM): A message transmitted by a Congestion Point to a Reaction Point, in response to a frame received from that Reaction Point, conveying congestion information used by the Reaction Point to reduce its transmission rate.

3.33 Congestion Notification Priority Value (CNPV): A value of the priority parameter that a congestion-aware system uses to support congestion notification.

3.34 Congestion Notification Tag (CN-TAG): A tag that conveys a Flow Identifier, that a Reaction Point can add to transmitted Congestion Controlled Flow frames, and that a Congestion Point includes in a Congestion Notification Message.

3.35 Congestion Point (CP): A VLAN-aware Bridge or end station Port function (31.1.1) that monitors a single queue serving one or more Congestion Notification Priority Values, can generate Congestion Notification Messages, and can remove Congestion Notification Tags.

3.36 congruent: Coinciding at all points when superimposed. In this standard the adjective congruent is used to describe two or more paths that can be taken by frames assigned to the same VLAN through bridges and LANs, and to describe the relationship between trees or sub-trees each composed of the paths that can be taken by a frame with a particular (source or destination) MAC address. Two paths are *reverse path congruent* if a unicast frame traverses exactly the same bridges and LANs along the path from the source to the destination, but in the reverse order as a frame transmitted from that destination to the source. If the path taken by a unicast frame to a given destination is the same as that taken by any multicast frame (assigned to the same VLAN) from that source to the same destination, those paths are *unicast multicast congruent*. If the path for a given frame relayed by a first bridge passes through a second bridge and is thereafter congruent with the path for any other frame (assigned to the same VLAN and destined to the same station) relayed by the second bridge, those paths are *downstream congruent*. A set of trees are said to be congruent if the paths they specify are congruent for all VLANs. See also *symmetric* (3.179).

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

3.38 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.

3.39 customer: The purchaser of a service instance from a service provider.

NOTE—Certain terms in this standard, including customer and service provider, reflect common business relationships that exist among the users of equipment implemented in conformance with IEEE Std 802.1Q. The use of these terms does not imply or impose any requirement on such business relationships. A customer is often a business.

3.40 Customer Backbone Port (CBP): A Backbone Edge Bridge Port that can receive and transmit I-tagged frames for multiple customers, and can assign B-VIDs and translate I-SID on the basis of the received I-SID.

3.41 Customer Bridge: A MAC Bridge as specified by IEEE Std 802.1D or a VLAN Bridge as specified by IEEE Std 802.1Q.

3.42 Customer Bridged Local Area Network: A network of Customer Bridges.

3.43 Customer Edge Port: A C-VLAN component Port on a Provider Edge Bridge that is connected to customer owned equipment and receives and transmits frames for a single customer.

3.44 customer equipment: The physical embodiment of one or more customer systems.

3.45 Customer MAC Address (C-MAC): A MAC address within a Customer Bridged Network (CBN) or Provider Bridged Network. Customer MAC Addresses are carried within an I-TAG in a Provider Backbone Bridged Network.

3.46 Customer Network Port: An S-VLAN component Port on a Provider Bridge or within a Provider Edge Bridge that receives and transmits frames for a single customer.

3.47 customer system: A system attached to Provider Bridged Network but not intended by the service provider to be under the control of the service provider.

3.48 customer tagged frames: Frames that include a C-TAG.

3.49 Customer VLAN: A VLAN identified by an C-VID.

3.50 Customer VLAN ID: A VLAN Identifier conveyed in a C-TAG.

3.51 Customer VLAN tag: A VLAN tag with a Tag Protocol Identification value allocated for “802.1Q Tag Protocol EtherType.”

3.52 data-driven and data-dependent connectivity fault management (DDCFM): Connectivity fault management capabilities to facilitate detecting and isolating data-driven and data-dependent faults in Virtual Bridged Local Area Networks.

3.53 data-driven and data-dependent fault (DDF): A fault that is sensitive to, or caused by, particular data patterns in frames transmitted by a service user.

3.54 Decapsulator Responder (DR): An entity for decapsulating Send Frame Messages and sending the decapsulated frames toward destinations specified by their own destination_address field.

3.55 Designated MSRP Node (DMN): A single station on a shared medium (e.g., IEEE 802.11 or a coordinated shared network (CSN)) that controls MSRP access for all other stations on that shared medium.

3.56 detagged frame: The detagged frame of an untagged frame is the frame itself. The detagged frame of a tagged frame or a priority-tagged frame is the frame that results from untagging the frame by the appropriate procedure.

3.57 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.

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

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

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

NOTE—See 6.8.

3.61 ESP-VID: A VID associated with a special value of the Multiple Spanning Tree Instance Identifier (MSTID) in the MST Configuration Table, the TE-MSTID, indicating that the VID is under the control of an external agent responsible for setting up Ethernet Switched Paths. Learning is disabled and forwarding is enabled for all frames allocated to ESP-VIDs.

3.62 Ethernet: A term that is used either to refer to the IEEE 802.3 media access method or to the IEEE 802.3 frame format.

NOTE—In many places, “Ethernet” is also used as part of a reserved term with its own specific meaning.

3.63 Ethernet Switched Path (ESP): A provisioned traffic engineered unidirectional connectivity path among two or more Customer Backbone Ports (CBPs) that extends over a PBBN. The path is identified by a 3-tuple <ESP-DA, ESP-SA, ESP-VID>, where ESP-DA and ESP-SA are MAC addresses and ESP-VID is a VID allocated to TE-MSTID. An Ethernet Switched Path is point-to-point or point-to-multipoint.

NOTE—The use of “Ethernet” in this definition does not imply that an ESP is limited to use in conjunction with IEEE802.3.

3.64 Expedited traffic: Traffic that requires preferential treatment as a consequence of jitter, latency, or throughput constraints, or as a consequence of management policy.

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

3.66 Flow Identifier (Flow ID): An identifier assigned by a congestion aware end station, unique within the scope of one or more of the source MAC addresses used by that system to transmit Congestion Controlled Flow frames, that can be used to associate each received Congestion Notification Message with the Reaction Point that rate controls the Congestion Controlled Flow that caused its transmission.

3.67 forced switch: An administrative command to force the backbone service instances assigned to a Traffic Engineering (TE) protection group to be carried by this group’s protection entity.

3.68 forward path test (FPT): A test to determine if a data frame that matches the specified filter condition can reach a specific location within a network.

3.69 Frame: A unit of data transmission on an IEEE 802 LAN that conveys a MAC Protocol Data Unit (MPDU) and can cause a service indication with, at a minimum, destination and source MAC addresses and an MAC Service Data Unit (MSDU) or an MPDU that is the result of a service request with those parameters.

3.70 Frame relay: Forwarding of frames between the Ports of a Bridge.

3.71 Group: A Group associates all of the following:

- a) A group MAC address
- b) A set of properties that define membership characteristics
- c) A set of properties that define the forwarding/filtering behavior of a Bridge with respect to frames destined for members of that group MAC address

with a set of end stations that all wish to receive information destined for that group MAC address. Members of such a set of end stations are said to be *Group members*.

A Group is said to *exist* if the properties associated with that Group are visible in an entry in the Filtering Database of a Bridge, or in the MRP state machines that characterize the state of the Group; a Group is said to *have members* if the properties of the Group indicate that members of the Group can be reached through specific Ports of the Bridge.

NOTE—An example of the information that Group members might wish to receive is a multicast video data stream.

3.72 I-component (I-comp): An S-VLAN component with one or more Provider Instance Ports (PIPs).

3.73 I-tagged frame: A frame that contains a Backbone Service Instance tag immediately following the Source MAC address field.

3.74 IEEE 802 Local Area Network (LAN): IEEE 802 LANs (also referred to as LANs) are LAN technologies that provide a MAC Service equivalent to the MAC Service defined in ISO/IEC 15802-1. IEEE 802 LANs include IEEE Std 802.3 (CSMA/CD), and IEEE Std 802.11 (Wireless) LANs.

3.75 Independent Virtual Local Area Network (VLAN) Learning (IVL): Configuration and operation of the Learning Process and the Filtering Database such that, for a given set of VIDs, if a given individual MAC Address is learned in association with one VID, that learned information is not used in forwarding decisions taken for that address relative to any other VID in the given set.

NOTE—In a Bridge that supports only IVL operation, the “given set of VIDs” is the set of all VIDs.

3.76 Independent Virtual Local Area Network (VLAN) Learning (IVL) Bridge: A Bridge that supports only Independent VLAN Learning.

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

3.78 Internal Spanning Tree (IST): The connectivity provided by the CIST within an MST Region.

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

NOTE—See 6.6.

3.80 latency: The delay experienced by a frame in the course of its propagation between two points in a network, measured from the time that a known reference point in the frame passes the first point to the time that the reference point in the frame passes the second point.

NOTE 1—Latency is sometimes referred to as *frame delay*.

NOTE 2—The term *frame* is used in this standard to refer to a data frame, and not to a video frame.

3.81 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.

3.82 Legacy region: A set of LANs connected such that there is physical connectivity between any pair of LANs using only IEEE 802.1D conformant, VLAN-unaware MAC Bridges.

NOTE—If, in a Bridged Local Area Network containing both IEEE 802.1D and IEEE 802.1Q Bridges, all IEEE 802.1Q Bridges were to be removed, the result would be one or more Bridged Local Area Networks, each with its own distinct Spanning Tree. Each of those networks is a legacy region.

3.83 Link status notification: Notification of the status of a LAN by transmitting a frame conveying information about the MAC_Operational parameter associated with the LAN.

3.84 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.

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

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

3.87 Listener: The end station that is the destination, receiver or consumer of a stream.

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

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

3.90 MAC status notification: A layer management interaction with an underlying MAC Service to change the status of the MAC_Operational parameter in a Bridge that is a peer user of that service.

3.91 MAC status propagation: The process of communicating a MAC_Operational parameter value through one or more TPMRs, using link status notification, MAC status notification, or both.

3.92 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.

3.93 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.

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

3.95 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.

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

3.97 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.

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

3.99 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.

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

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

3.102 manual switch: An administrative command to move the backbone service instances assigned to a Traffic Engineering (TE) protection group to a specific TE service instance (TESI) associated with this TE protection group in absence of signal failures on both of the entities in the group.

3.103 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.

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

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

3.106 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.

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

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

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

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

3.111 MST Bridge: A Bridge capable of supporting the CST, and one or more MSTIs, and of selectively mapping frames classified in any given VLAN to the CST or a given MSTI.

3.112 MST Configuration Table: A configurable table that allocates each and every possible VID to the Common Spanning Tree or a specific Multiple Spanning Tree Instance.

3.113 MST Region: One or more MST Bridges with the same MST Configuration Identifiers, interconnected by and including LANs for which one of those bridges is the Designated Bridge for the CIST and which have no bridges attached that cannot receive and transmit RST BPDUs.

3.114 Multicast:

- a) A group MAC address (noun).
- b) The transmission of a frame using a group MAC address as the destination address (verb).
- c) Containing a group MAC address as the destination address (adjective).

3.115 Multiple Spanning Tree Algorithm and Protocol (MSTP): The Multiple Spanning Tree Algorithm and Protocol described in Clause 13 of this standard.

3.116 Multiple Spanning Tree Bridge Protocol Data Unit (MST BPDU): The MST BPDU specified in Clause 14 of this standard.

3.117 Multiple Spanning Tree (MST) Configuration Identifier: A name for, revision level, and a summary of a given allocation of VLANs to Spanning Trees.

NOTE—Each MST Bridge uses a single MST Configuration Table and Configuration Identifier.

3.118 Multiple Spanning Tree Instance (MSTI): One of a number of Spanning Trees calculated by MSTP within an MST Region, to provide a simply and fully connected active topology for frames classified as belonging to a VLAN that is mapped to the MSTI by the MST Configuration Table used by the MST Bridges of that MST Region.

3.119 Multiple Stream Registration Protocol (MSRP): A protocol designed to provide Quality of Service for streams in bridged networks by reserving resources within each Bridge along the stream's paths.

3.120 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—Examples are ATM OAM {ITU-T I.610 (1999) [B40]}¹⁶ and IEEE 802.3 Clause 57 OAM.

3.121 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.

3.122 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).

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

3.124 PBB-TE Region: A contiguous set of IB-BEBs and BCBs, capable of providing TESIs, that have allocated a common subset of ESP-VIDs to an external agent that provides the active topology construction mechanism within this ESP-VID space and manages the Filtering Database of Bridges within the region to control the forwarding of frames with particular values of ESP-VID and destination MAC address.

¹⁶The numbers in brackets correspond to those of the bibliography in Annex M.

3.125 point-to-multipoint ESP: An ESP among one root Customer Backbone Port (CBP) and n leaf CBPs. The ESP-DA in the ESP's 3-tuple identifier is a group MAC address identifying the n leaf CBPs, and the ESP-SA is the individual MAC address of the root.

3.126 point-to-multipoint TESI: A TESI supported by a set of ESPs that comprises one point-to-multipoint ESP from a root to n leaves plus a point-to-point ESP from each of the leaves to the root.

NOTE—For reasons relating to TESI monitoring diagnostics, operational simplicity, etc. this standard assumes that the point-to-point ESPs associated with a point-to-multipoint TESI are routed from each of the n leaves to the root along the branches of the point-to-multipoint ESP. Support for a point-to-multipoint TESI that comprises ESPs not routed in this way is problematic and is not defined in this standard.

3.127 point-to-point ESP: An ESP between two Customer Backbone Ports (CBPs). The ESP-DA and the ESP-SA in the ESP's 3-tuple identifier are the individual MAC addresses of the two CBPs.

3.128 point-to-point TESI: A TESI supported by two point-to-point ESPs where the ESPs' endpoints have the same Customer Backbone Port (CBP) MAC addresses.

NOTE—For reasons relating to TESI monitoring diagnostics, operational simplicity, etc. this standard assumes that the point-to-point ESPs associated with a point-to-point TESI are symmetric. Support for a point-to-point TESI that comprises nonsymmetric ESPs is problematic and is not defined in this standard.

3.129 Port: A service access point and the interface stack supporting that service access point.

3.130 Port-based service interface: The interface provided by a Provider Bridge that associates all customer frames with a single service instance.

3.131 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.

3.132 Priority-tagged frame: A tagged frame whose tag header carries priority information but carries no VLAN identification information.

3.133 protocol group database: Specifies a group of protocols by assigning a unique protocol group identifier to all protocols of the same group.

3.134 protocol group identifier: Designates a group of protocols that are associated together when assigning a VID to a frame.

3.135 protocol template: A tuple of values that specify a data-link encapsulation format and an identification of the protocol layer above the data-link layer.

3.136 Provider Backbone Bridge (PBB): A Backbone Core Bridge or a Backbone Edge Bridge.

3.137 Provider Backbone Bridged Network (PBBN): A network using Backbone Edge Bridges and Backbone Core Bridges to interconnect Provider Bridged Networks and other networks.

3.138 Provider Bridge: An S-VLAN Bridge or a Provider Edge Bridge.

3.139 Provider Bridged Network: A network using Provider Bridges to offer customer protocol transparent connectivity.

3.140 Provider Edge Bridge: A system comprising a single S-VLAN component and one or more C-VLAN components implemented in accordance with Clause 5 of IEEE Std 802.1Q.

3.141 Provider Edge Port: A C-VLAN component Port within a Provider Edge Bridge that connects to a Customer Network Port and receives and transmits frames for a single customer.

3.142 Provider Instance Port (PIP): The set of Virtual Instance Ports that are supported by a single instance of the Internal Sublayer Service (ISS).

3.143 Provider Network Port: An S-VLAN component Port on a Provider Bridge that can transmit and receive frames for multiple customers.

3.144 Rapid Spanning Tree Algorithm and Protocol (RSTP): The Rapid Spanning Tree Algorithm and Protocol described in Clause 13.

3.145 Rapid Spanning Tree Bridge Protocol Data Unit (RST BPDU): The RST BPDU specified in 14.3.2.

3.146 Reaction Point (RP): An end station port function (31.2.2.2) that controls the transmission rate of frames for one or more Congestion Controlled Flows, receiving and using Congestion Notification Messages as part of determining that rate.

3.147 Reflected Frame Message (RFM): A CFM PDU transmitted by a Reflection Responder in order to perform forward path test.

3.148 Reflected Frame Message (RFM) Receiver: A function on a bridge port, an end station, or a test equipment to receive Reflected Frame Message(s) and pass them to the corresponding, nonstandardized, RFM analyzer.

3.149 Reflection Responder (RR): An entity that encapsulates and reflects selected data frames to a target location and allows a copy of each reflected frame to continue to its destination without encapsulation.

3.150 return path test (RPT): A test to determine if a data frame can be sent without error from a specific location within a network to a station specified by the destination_address field of the data frame.

3.151 Rooted-Multipoint: An association of Bridge Ports where each Bridge Port is designated as either a Root or a Leaf Port, and the service connectivity among the Bridge Ports is restricted such that an ingress frame at a Root Port may result in an egress frame at any or all Leaf Ports and other Root Ports, while an ingress frame at a Leaf Port may result in an egress frame at any or all Root Ports but never results in an egress frame at another Leaf Port (see 6.2.1, F.1.3.2).

3.152 S-tagged service interface: The interface provided by a Provider Bridge to allow the attached customer to select between and identify service instances using the S-VID associated with transmitted and received frames.

3.153 S-VLAN Bridge: A system comprising a single S-VLAN component implemented in accordance with Clause 5 of IEEE Std 802.1Q.

3.154 S-VLAN component: A VLAN-aware bridge component with each Port supported by an instance of the EISS that can recognize, insert, and remove Service VLAN tags.

3.155 Send Frame Message (SFM): A CFM PDU destined to a Decapsulator Responder in order to perform return path test.

3.156 Send Frame Message (SFM) Originator: A function on a bridge interface, an end station, or a test equipment to send SFM encapsulated data frames.

3.157 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.

3.158 service frame: A LAN frame exchanged between a provider and a customer.

3.159 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.

NOTE—In general, a service provider can use a number of technologies to support a service instance; within IEEE Std 802.1Q, a service instance is supported by a single S-VLAN.

3.160 service provider: An organization that contracts to provide one or more service instances to a customer.

3.161 service tagged frames: Frames that include an S-TAG.

3.162 Service VLAN: A VLAN identified by an S-VID.

3.163 Service VLAN ID: A VLAN Identifier conveyed in an S-TAG.

3.164 Service VLAN tag: A VLAN tag with a Tag Protocol Identification value allocated for “802.1Q Service Tag EtherType.”

3.165 Shared Virtual Local Area Network (VLAN) Learning (SVL): Configuration and operation of the Learning Process and the Filtering Database such that, for a given set of VLANs, if an individual MAC Address is learned in one VLAN, that learned information is used in forwarding decisions taken for that address relative to all other VLANs in the given set.

NOTE—In a Bridge that supports only SVL operation, the “given set of VLANs” is the set of all VLANs.

3.166 Shared Virtual Local Area Network (VLAN) Learning (SVL) Bridge: A type of Bridge that supports only Shared VLAN Learning.

3.167 Shared Virtual Local Area Network (VLAN) Learning (SVL)/Independent Virtual Local Area Network (VLAN) Learning (IVL) Bridge: An SVL/IVL Bridge is a type of Bridge that simultaneously supports both Shared VLAN Learning and Independent VLAN Learning.

3.168 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.

3.169 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.

3.170 Single Spanning Tree (SST) Bridge: A Bridge capable of supporting only a single spanning tree, the CST. The single spanning tree may be supported by the Spanning Tree Algorithm and Protocol (STP) defined in Clause 8 of IEEE Std 802.1D, 1998 Edition, or by the Rapid Spanning Tree Algorithm and Protocol (RSTP), defined in Clause 13.

3.171 Spanning Tree: A simply and fully connected active topology formed from the arbitrary physical topology of connected Bridged Local Area Network components by relaying frames through selected bridge ports and not through others. The protocol parameters and states used and exchanged to facilitate the calculation of that active topology and to control the MAC relay function.

3.172 Spanning Tree Algorithm and Protocol (STP): The Spanning Tree Algorithm and Protocol described in Clause 8 of IEEE Std 802.1D, 1998 Edition.

3.173 Spanning Tree Bridge Protocol Data Unit (ST BPDU): A Bridge Protocol Data Unit specified for use by the Spanning Tree Algorithm and Protocol, i.e., a Configuration or Topology Change Notification BPDU as described in Clause 9 of IEEE Std 802.1D-2004.

3.174 Stream: A unidirectional flow of data (e.g., audio and/or video) from a Talker to one or more Listeners.

3.175 stream reservation (SR) class: A traffic class whose bandwidth can be reserved for AV traffic. A priority value is associated with each SR class. SR classes are denoted by consecutive letters of the alphabet, starting with A and continuing for up to seven classes.

3.176 Stream Reservation Protocol (SRP) domain boundary port: A Port of a station that is a member of an SRP domain for a given SR class, and that is connected via an individual LAN to a Port of a station that either is within a different SRP domain for that SR class, or is not part of any SRP domain.

NOTE—The term *SRP domain* is defined in 6.6.4.

3.177 Stream Reservation Protocol (SRP) domain core port: A Port of a station that is a member of an SRP domain for a given SR class, and that is connected, via an individual LAN that is part of the active topology, to a Port of a station that is a member of the same SRP domain for that SR class.

NOTE—The term *SRP domain* is defined in 6.6.4.

3.178 StreamID: A 64-bit field that uniquely identifies a stream.

3.179 symmetric: In this standard the adjective symmetric is used to describe paths, trees or sets of trees. Symmetric algorithms are used to create a set of trees that are *reverse path congruent* (3.36) with respect to sources and destinations. Symmetric algorithms are used to create a set of trees that are *unicast multicast congruent* (3.36) with respect to a given source.

3.180 T-component: A TPMR component with one Provider Instance Port (PIP).

3.181 Tag header: A *tag header* allows priority information, and optionally, VLAN identification information, to be associated with a frame.

3.182 Tagged frame: A *tagged frame* is a frame that contains a tag header immediately following the Source MAC Address field of the frame.

3.183 Talker: The end station that is the source or producer of a stream.

3.184 time-sensitive stream: A stream of data frames that are required to be delivered with a bounded latency.

3.185 TPMR component: A VLAN-unaware bridge component with two Bridge Ports supported by an instance of the ISS as specified in 5.13.

3.186 Traffic class: Traffic classes are numbered from zero through N-1, where N is the number of outbound queues associated with a given Bridge Port, and $1 \leq N \leq 8$, and each traffic class has a one-to-one correspondence with a specific outbound queue for that Port. Traffic class 0 corresponds to nonexpedited traffic; nonzero traffic classes correspond to expedited classes of traffic. A fixed mapping determines, for a given priority associated with a frame and a given number of traffic classes, what traffic class will be assigned to the frame.

3.187 Traffic Engineering service instance (TESI): An instance of the MAC service supported by a set of ESPs and identified by TE-SID, forming a bidirectional service. A TESI is point-to-point or point-to-multipoint.

3.188 Traffic Engineering service instance identifier (TE-SID): An identifier of the TESI which corresponds to a series of 3-tuples <ESP-DA, ESP-SA, ESP-VID>, each one identifying one of the TESI's ESPs.

NOTE—The TE-SID is not used as a tag parameter.

3.189 Traffic Engineering (TE): The process that controls the placement of traffic demands in a network, in order to optimize the resource utilization and to ensure that the quality of service objectives for each defined class of service are met.

3.190 Traffic Engineering (TE) protection group: A group of two point-to-point TESIs between a pair of Customer Backbone Ports (CBPs), which carries an assigned set of backbone service instances, and continues to carry these backbone service instances if any one of the TESIs in the group is failed or disabled.

3.191 Traffic Specification (TSpec): A specification that characterizes the bandwidth that a stream can consume.

3.192 Two-Port Media Access Control (MAC) Relay (TPMR): A bridge that has exactly two externally accessible Ports, and supports a subset of the functionality of a MAC Bridge as specified in 5.14.

3.193 Unicast:

- a) An individual MAC address (noun).
- b) The transmission of a frame using an individual MAC address as the destination address (verb).
- c) Containing an individual MAC address as the destination address (adjective).

3.194 Untagged frame: An *untagged frame* is a frame that does not contain a tag header immediately following the Source MAC Address field of the frame.

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

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

3.197 Virtual Bridged Local Area Network: A concatenation of individual IEEE 802 LANs interconnected by Bridges, including VLAN-aware Bridges.

3.198 Virtual Instance Port (VIP): A Bridge Port on an I-component or a T-component in a Backbone Edge Bridge that provides access to a single backbone service instance.

3.199 VLAN: The closure of a set of MAC SAPs such that a data request in one MAC SAP in the set is expected to result in a data indication in another MAC SAP in the set.

NOTE—See also 6.2.

3.200 VLAN-aware Bridge: A Bridge that recognizes frames with a VLAN tag and can insert or remove tag headers.

3.201 VLAN-aware bridge component: The media access method independent functionality supporting an instance of the EISS at each Port that can recognize, insert, and remove VLAN tags, and the functionality that relays frames between Ports.

3.202 VLAN Bridge: A system comprising a single C-VLAN component implemented in accordance with Clause 5 of IEEE Std 802.1Q.

3.203 VLAN-tagged frame: A *VLAN-tagged frame* is a tagged frame whose tag header carries both VLAN identification and priority information.

3.204 VLAN-unaware Bridge: A Bridge that does not recognize VLAN-tagged frames.

4. Abbreviations

The following abbreviations are used in this standard:

AV	audio/video
AVB	audio/video bridging
B-BEB	B type Backbone Edge Bridge
B-Comp	B-component
B-DEI	Backbone VLAN Drop Eligible Indicator
B-DA	Backbone Destination MAC address
B-MAC	Backbone MAC address
B-PCP	Backbone VLAN Priority Code Point
B-SA	Backbone Source MAC address
B-TAG	Backbone VLAN tag
B-VID	Backbone VLAN ID
B-VLAN	Backbone VLAN
BCB	Backbone Core Bridge
BEB	Backbone Edge Bridge
BNF	Backus-Naur Form
BPDU	Bridge Protocol Data Unit (IEEE Std 802.1D)
BSI	Backbone service instance
C-DA	Customer Destination MAC address
C-MAC	Customer MAC address
C-SA	Customer Source MAC address
C-TAG	Customer VLAN tag
C-VID	Customer VLAN Identifier
C-VLAN	Customer VLAN
CB	Customer Bridge
CBN	Customer Bridged Network
CBP	Customer Backbone Port
CCF	Congestion Controlled Flow
CCM	Continuity Check Message
CE	Customer Equipment
CFM	Connectivity Fault Management
CIST	Common and Internal Spanning Tree
CIST-MSTID	Common and Internal Spanning Tree Multiple Spanning Tree Instance Identifier
CN-TAG	Congestion Notification Tag
CN	Congestion Notification
CND	Congestion Notification Domain
CNM	Congestion Notification Message
CNPV	Congestion Notification Priority Value
CP	Congestion Point
CPID	Congestion Point Identifier
CSN	Coordinated Shared Network
CST	Common Spanning Tree
DDCFM	data-driven and data-dependent connectivity fault management
DDF	data-driven and data-dependent fault
DEI	Drop Eligible Indicator
DMN	Designated MSRP Node
DoSAP	Domain Service Access Point
DR	Decapsulator Responder
EISS	Enhanced Internal Sublayer Service (6.8)
ESP	Ethernet Switched Path

EUI-48	48-bit Extended Unique Identifier ¹⁷
Fb	Quantized Feedback
FCS	Frame Check Sequence
FDB	Filtering Database
FID	Filtering Identifier (8.8.8, 8.9.3)
Flow ID	Flow Identifier
FPT	forward path test
FQTSS	Forwarding and Queuing Enhancements for time-sensitive streams
FS	Forced Switch
I-BEB	I type Backbone Edge Bridge
I-Comp	I-component
I-DEI	Backbone Service Instance Drop Eligible Indictor
I-PCP	Backbone Service Instance priority code point
I-TAG	Backbone Service Instance tag
I-SID	Backbone Service Instance Identifier
ISAP	Intermediate Service Access Point
ISS	Internal Sublayer Service (6.4 of IEEE Std 802.1D-2004)
IST	Internal Spanning Tree
ITB-BEB	ITB type Backbone Edge Bridge
ITU-T	International Telecommunications Union—Telecommunication Standardization Sector
IVL	Independent VLAN Learning (3.75)
LACP	Link Aggregation Control Protocol
LAG	Link Aggregation Group
LAN	Local Area Network (IEEE Std 802)
LBM	Loopback Message
LBR	Loopback Reply
LLC	Logical Link Control (ISO/IEC 8802-2)
LMI	Layer Management Interface
LOM	Linktrace Output Multiplexer
LoP	Lockout of Protection
LSB	Least Significant Bit
LTM	Linktrace Message
LTR	Linktrace Reply
MA	Maintenance Association
MAC	Medium Access Control (IEEE Std 802)
MAD	MRP Attribute Declaration
MAID	Maintenance Association Identifier
MAP	MRP Attribute Propagation
MBWA	Mobile Broadband Wireless Access Method
MCID	MST Configuration Identifier
ME	Maintenance Entity
MEP	Maintenance association End Point
MEPID	Maintenance association End Point Identifier
MHF	MIP Half Function
MIB	Management Information Base (Clause 17)
MIP	Maintenance domain Intermediate Point
MMRP	Multiple MAC Registration Protocol
MMRPDU	Multiple MAC Registration Protocol Data Unit
MP	Maintenance Point
MRP	Multiple Registration Protocol
MRPDU	Multiple Registration Protocol Data Unit
MS	MAC Service

¹⁷A tutorial on the structure and use of EUI-48 identifiers can be found at <http://standards.ieee.org/develop/regauth/tut/eui.pdf>

MSAP	MAC Service Access Point
MSB	Most Significant Bit
MSDU	MAC Service Data Unit (ISO/IEC 15802-1)
MSP	MAC status protocol
MSPDU	MAC Status Protocol Data Unit
MSPE	MAC Status Propagation Entity
MSRP	Multiple Stream Registration Protocol
MSRPDU	Multiple Stream Registration Protocol Data Unit
MSS	MAC Status Shim
MST	Multiple Spanning Tree
MST BPDU	Multiple Spanning Tree Bridge Protocol Data Unit
MSTI	Multiple Spanning Tree Instance
MSTP	Multiple Spanning Tree Protocol
MVRP	Multiple VLAN Registration Protocol
MVRPDU	Multiple VLAN Registration Protocol Data Unit
PB	Provider Bridge
PBB-TE	Provider Backbone Bridge-Traffic Engineering
PBB	Provider Backbone Bridge
PBBN	Provider Backbone Bridged Network
PBN	Provider Bridged Network
PCP	Priority Code Point
PDU	Protocol Data Unit
PICS	Protocol Implementation Conformance Statement (Annex A)
PID	Protocol IDentifier
PIP	Provider Instance Port
PVID	Port VID
QCN	Quantized Congestion Notification protocol
QoS	quality of service
RED	Random Early Detection
RFM	Reflected Frame Message
RP	Reaction Point
RPR	Resilient Packet Ring
RPT	return path test
RR	Reflection Responder
RST BPDU	Rapid Spanning Tree Bridge Protocol Data Unit
RSTP	Rapid Spanning Tree Protocol
S-TAG	Service VLAN tag
S-VID	Service VLAN Identifier
S-VLAN	Service VLAN
SF	Signal Fail
SFM	Send Frame Message
SNM	Status Notification Machine
SR_PVID	Stream Reservation Port VLAN Identifier
SR	stream reservation
SRP	Stream Reservation Protocol
SST	Single Spanning Tree
ST BPDU	Spanning Tree Bridge Protocol Data Unit
STM	Status Transition Machine
STP	Spanning Tree Protocol
STPID	SNAP-encoded Tag Protocol Identifier (9.3)
SVL	Shared VLAN Learning (3.165)
TC	Textual Conventions
TCI	Tag Control Information (9.3)
TE-MSTID	Traffic Engineering Multiple Spanning Tree Instance Identifier

TE-SID	Traffic Engineering service instance identifier
TESI	Traffic Engineering service instance
TPID	Tag Protocol Identifier (9.3)
TPMR	Two-Port Media Access Control (MAC) Relay
TSpec	traffic specification
VID	VLAN Identifier (7.2, 9.3)
VIP	Virtual Instance Port
VLAN	Virtual LAN
WRED	Weighted Random Early Detection

5. Conformance

This clause specifies the mandatory and optional capabilities provided by conformant implementations of this standard. An implementation can

- a) Compose all or part of the functionality of a system;
- b) Provide, as specified by this standard, one or more instances of the MAC Service to other functional entities whose specification is outside the scope of this standard;
- c) Provide, as specified by this standard, one or more instances of the MAC Internal Sublayer Service (ISS) to other implementations or instances of the same implementation that conform to this standard.

Accordingly, and as detailed in 5.4, this clause specifies conformance requirements for common systems and for functional components within systems, possibly connected to other system components with interfaces that are not otherwise accessible.

5.1 Requirements terminology

For consistency with existing IEEE and IEEE 802.1 standards, requirements placed upon conformant implementations of this standard are expressed using the following terminology:

- a) **Shall** is used for mandatory requirements;
- b) **May** is used to describe implementation or administrative choices (“may” means “is permitted to,” and hence, “may” and “may not” mean precisely the same thing);
- c) **Should** is used for recommended choices (the behaviors described by “should” and “should not” are both permissible but not equally desirable choices).

NOTE—The usage in 17.7 follows the IETF model.

The PICS proformas (see Annex A for Bridges and Annex B for end station implementations) reflect the occurrences of the words “shall,” “may,” and “should” within the standard.

The standard avoids needless repetition and apparent duplication of its formal requirements by using **is**, **is not**, **are**, and **are not** for definitions and the logical consequences of conformant behavior. Behavior that is permitted but is neither always required nor directly controlled by an implementer or administrator, or whose conformance requirement is detailed elsewhere, is described by **can**. Behavior that never occurs in a conformant implementation or system of conformant implementations is described by **cannot**. The word **allow** is used as a replacement for the cliche “Support the ability for,” and the word **capability** means “can be configured to.”

5.2 Conformant components and equipment

This subclause specifies requirements and options for the following core component:

- a) VLAN-aware Bridge component (5.4);
- b) VLAN-unaware Bridge component (5.12);

for the following components that use that core functionality:

- c) C-VLAN component (5.5);
- d) S-VLAN component (5.6);
- e) I-component (5.7);

- f) B-component (5.8);
- g) TPMR component (5.13);
- h) T-component (5.15);

and for the following systems that include instances of the above components:

- i) VLAN Bridge (5.9);
- j) S-VLAN Bridge (5.10.1);
- k) Provider Edge Bridge (5.10.2);
- l) Backbone Edge Bridge (5.11);
- m) TPMR (5.14).

NOTE—A VLAN Bridge can also be referred to as a Customer Bridge or a C-VLAN Bridge. Both S-VLAN Bridges and Provider Edge Bridges are examples of Provider Bridges.

5.3 Protocol Implementation Conformance Statement (PICS)

A claim of conformance specifies implementation of a C-VLAN component, an S-VLAN component, an I-component, a B-component, or a specific system. A component or system can support multiple claims for a range of possible behaviors.

The supplier of an implementation that is claimed to conform to this standard shall provide the information necessary to identify both the supplier and the implementation, and shall complete a copy of the PICS proforma provided in Annex A for that specific Bridge component or system, or Annex I for an end station implementation, together with any further information and completed PICS(s) required to identify subcomponents.

NOTE 1—C-VLAN component and S-VLAN component PICS both require completion of a PICS for a VLAN-aware bridge component; the VLAN Bridge PICS requires a claim of conformance for a single C-VLAN component; the Provider Edge Bridge requires a claim of conformance for a single S-VLAN component and one or more claims for C-VLAN components.

NOTE 2—The claim of conformance that could be made to IEEE Std 802.1Q for an implementation of a VLAN-aware Bridge is replaced by a claim of conformance to a VLAN Bridge. Although the current and subsequent amendment(s) or revision(s) of this standard have changed the presentation of the information, the technical requirements of conformance remain unchanged.

NOTE 3—I-component and B-component PICS both require completion of a PICS for an S-VLAN component.

5.4 VLAN-aware Bridge component requirements

An implementation of a VLAN-aware Bridge component shall

- a) Conform to the requirements of IEEE Std 802.1D, as modified by the provisions of this standard;
- b) Conform to the relevant standard for the Media Access Control technology implemented at each Port in support of the MAC ISS, as specified in 6.6 and 6.7;
- c) Support the MAC Enhanced Internal Sublayer Service at each Port, as specified in 6.8 and 6.9;
- d) Implement an ISO/IEC 8802-2 conformant LLC class with Type 1 operation as required by 8.2;
- e) Relay and filter frames as described in 8.1 and specified in 8.5, 8.6, 8.7, and 8.8;
- f) On each Port, support at least one of the permissible values for the Acceptable Frame Types parameter, as defined in 6.9;
- g) Support the following on each Port that supports untagged and priority-tagged frames:
 - 1) A Port VLAN Identifier (PVID) value (6.9);
 - 2) Configuration of at least one VID whose untagged set includes that Port (8.8.2);

- 3) Configuration of the PVID value via management operations (12.10);
- 4) Configuration of Static Filtering Entries via management operations (12.7).
- h) Allow tag headers to be inserted, modified, and removed from relayed frames, as specified in 8.1 and Clause 9, as required by the value(s) of the Acceptable Frame Types parameter supported on each Port, and by the ability of each Port to transmit VLAN-tagged and/or untagged frames;
- i) Allow automatic configuration and management of VLAN topology using the Multiple VLAN Registration Protocol (MVRP) (5.4.2, Clause 11) on all Ports;
- j) Allow static and dynamic configuration information for at least one VID, by means of Static and Dynamic VLAN Registration Entries in the Filtering Database (8.11);
- k) Support at least one Filtering Identifier (FID) (6.6, 8.8.3, 8.8.8, and 8.8.9);
- l) Allow allocation of at least one VID to each FID that is supported (6.6, 8.8.3, 8.8.8, and 8.8.9).

NOTE—Under some circumstances, the ability for VLAN-aware Bridges to successfully interoperate depends on the number of FIDs supported and on the number of VIDs that can be allocated to each FID. These circumstances are discussed in Annex B, along with interoperability implications.

5.4.1 VLAN-aware Bridge component options

An implementation of a VLAN-aware Bridge component may

- a) Support MST operation (5.4.1.1);
- b) Support Port-and-Protocol-based VLAN classification (5.4.1.2), including multiple VID values per port, administrative control of the values of the multiple VIDs, and a Protocol Group Database.
- c) Support CFM operation (5.4.1.4);
- d) Support congestion notification (5.4.3).
- e) Support Extended Filtering Services (6.16.5) and the operation of Multiple MAC Registration Protocol (MMRP, see 10.9) to support automatic configuration and management of MAC address information on all Ports (5.4.1.3);
- f) Allow the Filtering Database to contain Static and Dynamic VLAN Registration Entries (8.8) for more than one VID, up to a maximum of 4094 VIDs;

NOTE—The maximum number of VIDs that can be supported is 4094 rather than 4096, as the VID values 0 and FFF are reserved, as indicated in Table 9-2. As conformance to this standard is only with regard to externally visible protocol behavior, this limit on the number of VIDs that can be supported does not imply any such limitation with regard to the internal architecture of a Bridge.

- g) On each Port, support all of the permissible values for the Acceptable Frame Types parameter, as defined in 8.3, and support configuration of the parameter value via management;
- h) Support enabling and disabling of Ingress Filtering (6.9);
- i) Allow configuration of more than one VID whose untagged set includes that Port (8.8.2);
- j) Support the management functionality defined in Clause 12;
- k) Support more than one FID (8.8);
- l) Allow allocation of more than one VID to each supported FID (8.8, 8.8.8);
- m) Allow configuration of VLAN Learning Constraints (8.8.8, 12.10.3);
- n) Allow configuration of fixed VID to FID allocations (8.8.8, 12.10.3);
- o) Allow configuration of the Restricted_MAC_Address_Registration parameter (10.12.2.3) for each Port of the Bridge;
- p) Support the ability to configure the value of the Restricted_VLAN_Registration parameter (11.2.3.2.3) for each Port of the Bridge.
- q) Support forwarding and queuing for time-sensitive streams (5.4.1.5).
- r) Support SMIv2 MIB modules for the management of VLAN-aware Bridge capabilities (Clause 17).
- s) Support Multiple Stream Reservation Protocol (MSRP; Clause 35).

NOTE—In previous versions of IEEE Std 802.1Q, the MIB modules were maintained by IETF (e.g., IETF RFC 4363 for the Q-BRIDGE MIB). For this standard, the relevant set of MIB modules required to manage a Bridge is contained in Clause 17.

5.4.1.1 Multiple Spanning Tree (MST) operation (optional)

A VLAN-aware Bridge implementation in conformance to the provisions of this standard for an MST Bridge (5.4.1, 8.3, 8.4, 8.6.1, 8.9, 8.10, 11.2, 11.2.3.1, Clause 13, and Clause 14) shall:

- 1) Support the Multiple Spanning Tree Protocol (MSTP) as specified in Clause 13;
- 2) Support the Common and Internal Spanning Tree (CIST) plus a stated maximum number of Multiple Spanning Tree Instances (MSTIs), where that number is at least 2 (8.9) and at most 64 (13.13);

NOTE—In other words, a conformant MST Bridge supports a minimum of three spanning tree instances—the CIST and at least two additional MSTIs.

- 3) Support a stated maximum number of FIDs not less than the number of MSTIs (8.9);
- 4) Support the ability to associate each FID to a spanning tree (8.9.3);
- 5) Support the transmission and reception of MST Configuration Identifier information (8.9.2);
- 6) Support a Port State for each Port for each spanning tree instance supported (8.4, 13.36);
- 7) Support operation of spanning tree protocol for each spanning tree instance and Port (8.10, Clause 13);
- 8) Use the Bridge Group Address as specified in 8.13.3;
- 9) Support the default values for Bridge Forward Delay and Bridge Priority parameters specified in 13.24;
- 10) Support the operation of MVRP in each supported spanning tree context (11.2.3.1.1, 11.2.3.1.2);
- 11) Support the Bridge management functions for the bridge protocol entity for each supported spanning tree, independently (12.8);
- 12) Support, in particular, management of the bridge priority parameters, and of the port priority and path cost parameters for every port, independently for each supported spanning tree (12.8.1.1, 12.8.1.3, 13.25);
- 13) Support VLAN management functions for each supported spanning tree (12.10.1 and 12.11.1);
- 14) Support management of the MSTI configuration (12.12).

A VLAN-aware Bridge implementation in conformance to the provisions of this standard for an MST Bridge (5.4.1, Clause 13) may

- 15) Support a greater number of FIDs than spanning trees (8.8.8).
- 16) Support SMIv2 MIB modules for the management of VLAN-aware Bridge capabilities (Clause 17).

5.4.1.2 Port-and-Protocol-based VLAN classification (optional)

A VLAN-aware bridge component implementation in conformance to the provisions of this standard for Port-and-Protocol-based VLAN classification (5.4.1) shall

- 1) Support one or more of the following Protocol Classifications and Protocol Template formats: Ethernet, RFC_1042, SNAP_8021H, SNAP_Other, or LLC_Other (6.12);

and may

- 2) Support configuration of the contents of the Protocol Group Database.

5.4.1.3 Multiple MAC Registration Protocol (MMRP) operation (optional)

A VLAN-aware Bridge implementation in conformance to the provisions of this standard for the support of MMRP shall

- a) Conform to the operation of the MRP Applicant, Registrar, LeaveAll, and Periodic Transmission state machines, as defined in 10.7.7, 10.7.8, 10.7.9, and 10.7.10, as required for one or the other of the following variants of the MRP Participant, as defined in 10.6 and 10.7:
 - 1) The Full Participant.
 - 2) The Full Participant, point-to-point subset.
- b) Exchange MRPDUs as required by those state machines, formatted in accordance with the generic PDU format described in 10.8, and able to carry application-specific information as defined in 10.12, using the group MAC addresses reserved for use by MRP applications, as defined in Table 10-1.
- c) Implement the MMRP Application component as defined in 10.12.
- d) Propagate registration information in accordance with the operation of MAP for the Base Spanning Tree Context, as specified in 10.3.1.
- e) Forward, filter or discard MAC frames carrying any MRP Application address as the destination MAC Address in accordance with the requirements of 8.13.6.

5.4.1.4 Connectivity Fault Management (optional)

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

- a) Support the creation of Maintenance Domains at eight MD Levels, with multiple nonoverlapping Maintenance Domains at each MD Level, using the managed objects specified in 12.14;
- b) Support the creation of a Maintenance Association (MA) for each VID 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 for each VID on each Port, each MEP at a different MD Level;
- e) Support the creation of eight Down MEPs for each VID on each Port, each MEP at a different MD Level;
- f) Support the creation of eight Down MEPs attached to no VID 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 MD Levels lower than or equal to 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 VID (see 22.1.5).
- p) Support the creation and operation of Reflection Responder and RFM Receiver for the forward path test.
- q) Support the creation and operation of Decapsulator Responder and SFM Originator for the return path test.

5.4.1.5 Forwarding and queuing for time-sensitive streams—requirements

A VLAN-aware Bridge component implementation that conforms to the provisions of this standard for forwarding and queuing for time-sensitive streams shall:

- a) Support a minimum of two traffic classes on all Ports, of which
 - 1) A minimum of one traffic class supports the strict priority algorithm for transmission selection (8.6.8.1), and
 - 2) One traffic class is an SR class.
- b) Support the operation of the credit-based shaper algorithm (8.6.8.2) on all Ports as the transmission selection algorithm used for the SR class.
- c) Support SRP domain boundary port priority regeneration override as defined in 6.9.4, and the default priority regeneration override value defined in Table 6-6, for SR class “B.”
- d) Support the tables and procedures for mapping priorities to traffic classes as defined in 34.5.

A VLAN-aware Bridge component implementation that conforms to the provisions of this standard for forwarding and queuing for time-sensitive streams may:

- e) Support the management entities defined in 12.20 by means of the SNMPv2 MIB module defined in 17.7.12.
- f) Support two or more SR classes (a maximum of seven), and support the operation of the credit-based shaper algorithm (8.6.8.2) on all Ports as the transmission selection algorithm used for those SR classes. The number of SR classes supported shall be stated in the PICS.
- g) Support SRP domain boundary port priority regeneration override as defined in 6.9.4, and the default priority regeneration override value defined in Table 6-6, for SR class “A.” If more than two SR classes are supported, the default priority regeneration override values used for the additional SR classes shall be stated in the PICS.

5.4.2 Multiple VLAN Registration Protocol (MVRP) requirements

A VLAN-aware Bridge implementation in conformance to the provisions of this standard for the support of MVRP shall

- a) Conform to the operation of the MRP Applicant, Registrar, LeaveAll, and Periodic Transmission state machines, as defined in 10.7.7, 10.7.8, 10.7.9, and 10.7.10, as required for one or the other of the following variants of the MRP Participant, as defined in 10.6 and 10.7:
 - 1) The Full Participant.
 - 2) The Full Participant, point-to-point subset.
- b) Exchange MRPDUs as required by those state machines, formatted in accordance with the generic PDU format described in 10.8, and able to carry application-specific information as defined in 11.2, using the group MAC addresses reserved for use by MVRP.
- c) Implement the MVRP Application component as defined in 11.2.
- d) Propagate registration information
 - 1) In an SST bridge, in accordance with the operation of MAP for the Base Spanning Tree Context, as specified in 10.3 and 10.3.1; or
 - 2) in an MST bridge, in accordance with the operation of MAP for multiple Spanning Tree contexts as specified in 11.2.3.1.2 of this standard.
- e) Forward, filter or discard MAC frames carrying any MRP Application address as the destination MAC Address in accordance with the requirements of 8.13.6.
- f) If a VID Translation Table (6.9) is in use for a Bridge Port, translate VIDs in MVRPDUs as specified in 11.2.4.

5.4.3 VLAN-aware Bridge requirements for congestion notification

A VLAN-aware Bridge implementation that conforms to the provisions of this standard for congestion notification (Clause 30 through Clause 33) shall

- a) Support, on one or more Ports, the creation of at least one Congestion Point (32.1.1);
- b) Support, at each Congestion Point, the generation of Congestion Notification Messages (32.7) and the removal of Congestion Notification Tags (32.1.1);
- c) Support the ability to configure the variables controlling the operation of each Congestion Point (12.21.1, 12.21.2, 12.21.3, 12.21.4);
- d) Support the operation of each Port in each of the Congestion Notification Domain defense modes separately for each Congestion Notification Priority Value (32.1.1);
- e) Conform to the required capabilities of the Link Layer Discovery Protocol (LLDP) of 5.2 of IEEE Std 802.1AB-2009;
- f) Support the use of the Congestion Notification TLV in LLDP (32.1.1).

A Provider Instance Port (PIP, 6.10) that conforms to the provisions of this standard for congestion notification shall

- g) Support Congestion Notification Message translation (32.16).

A Provider Edge Port (15.4) that conforms to the provisions of this standard for congestion notification shall

- h) Support Congestion Notification Message translation (32.16).

A VLAN-aware Bridge implementation that conforms to the provisions of this standard for congestion notification may

- i) Support the creation of up to seven Congestion Points on a Bridge Port (31.1.1); and/or
- j) Support the IEEE8021-CN-MIB (17.7.13).

5.4.4 Multiple Stream Registration Protocol (MSRP) requirements

A VLAN-aware Bridge implementation in conformance to the provisions of this standard for the support of MSRP shall:

- a) Conform to the operation of the MRP Applicant, Registrar, and LeaveAll state machines, as defined in 10.7.7, 10.7.8, and 10.7.9, as required for one or more of the following variants of the MRP Participant, as defined in 10.6 and 10.7:
 - 1) The Full Participant.
 - 2) The Full Participant, point-to-point subset.
- b) Exchange MRPDUs as required by those state machines, formatted in accordance with the generic PDU format described in 10.8, and able to carry application-specific information as defined in Clause 35, using the group MAC addresses reserved for use by MRP applications, as defined in Table 10-1.
- c) Implement the MSRP Application component as defined in Clause 35.
- d) Propagate registration information in accordance with the operation of MAP for the Base Spanning Tree Context, as specified in 10.3.1.
- e) Forward, filter or discard MAC frames carrying any MRP Application address as the destination MAC Address in accordance with the requirements of 8.13.6.
- f) Not redeclare its attributes unless responding to a LeaveAll event.

NOTE—In order to meet the requirement specified in 5.4.4(f), the Periodic Transmission state machine (10.7.10) is specifically excluded from MSRP. A Bridge is allowed to generate an MSRP LeaveAll event to force an immediate redeclaration of all MSRP attributes from its neighbor(s).

5.5 C-VLAN component conformance

A C-VLAN component comprises a VLAN-aware Bridge component with the EISS on all Ports supported by the use of a Customer VLAN tag (C-TAG) (6.8, 9.5).

A conformant implementation of a C-VLAN component shall

- a) Comprise a single conformant VLAN-aware bridge component;
- b) Recognize and use C-TAGs;
- c) Filter the Reserved MAC Addresses specified in Table 8-1;
- d) Use the Customer Bridge MVRP Address specified in Table 10-1; and

shall not

- e) Use a VID Translation Table (6.9) on any Port;
- f) Use Service VLAN tags (S-TAG) except in support of the functionality specified in 6.13.

5.5.1 C-VLAN component options

A conformant C-VLAN component may implement the options specified for a VLAN-aware bridge component whose use is not specifically prohibited by 5.4.1 and may

- a) Support encoding the drop_eligible parameter in the PCP field of the tag header (6.9.3); and
- b) Allow support of the ISS as specified in 6.13 to facilitate priority selection on a Provider Bridged Network; and
- c) Implement RSTP, with the enhancements to support Customer Edge Ports, as specified in 13.39.

5.6 S-VLAN component conformance

An S-VLAN component comprises a VLAN-aware Bridge component with the EISS on all Ports supported by the use of a Service VLAN tag (S-TAG) (6.8, 9.5).

A conformant implementation of an S-VLAN component shall

- a) Comprise a single conformant VLAN-aware bridge component; and
- b) Recognize and use Service VLAN tags;
- c) Support encoding the drop_eligible parameter in the PCP field of the tag header (6.9.3);
- d) Filter the Reserved MAC Addresses specified in Table 8-2;
- e) Use the Provider Bridge MVRP Address specified in Table 8-1;
- f) Allow the Acceptable Frame Types parameter (6.9) to be set to *Admit All Frames* for each Port;
- g) Allow the Enable Ingress Filtering parameter (8.6.2) to be set for each Port; and

shall not

- h) Recognize or use Customer VLAN tags;
- i) Allow support of the ISS as specified in 6.13 for any of its Ports;
- j) Configure the Customer Bridge MVRP Addresses specified in Table 10-1 in the Filtering Database (8.8) or Permanent Database (8.8.11);
- k) Use the Customer Bridge MVRP Address specified in Table 10-1.

5.6.1 S-VLAN component options

A conformant S-VLAN component may implement any of the options specified for a VLAN-aware Bridge component (5.4.1) and may

- a) Allow translation of VIDs through support of a VID Translation Table (6.9) on each Port.

5.6.2 S-VLAN component requirements for PBB-TE

An S-VLAN component implementation that conforms to the provisions of this standard for PBB-TE (25.10) shall

- a) Disable learning for a set of B-VIDs allocated to TE-MSTID as specified in 8.4 and in 8.9; and
- b) Discard frames with unregistered destination addresses for B-VIDs allocated to TE-MSTID (8.8.1).

An S-VLAN component implementation that conforms to the provisions of this standard for PBB-TE (25.10) may

- c) Allow control of all B-VIDs by an external agent, conforming to the Reserved VID values of Table 9-2; and/or
- d) Support MIP creation and the corresponding CFM extensions for TESIs as specified in 26.9.

5.7 I-component conformance

An I-component comprises an S-VLAN component (5.6) with the EISS on each Customer Network Port supported by the use of a Service VLAN tag (6.9, 9.5), and the EISS for each Virtual Instance Port configured on a Provider Instance Port supported by the use of both a Service VLAN tag (6.9, 9.5) and a Backbone Service Instance tag (I-TAG) (6.10, 9.5).

A conformant implementation of an I-component shall

- a) Comprise a single conformant S-VLAN component;
- b) Recognize and use Backbone Service Instance tags (I-TAGs) on one or more Provider Instance Ports (6.10);
- c) Support 1:1 mapping between S-VID values and I-SID values;
- d) Support the connection_identifier parameter on the EISS of each Virtual Instance Port (6.10) and in the Filtering Database (8.8.12);
- e) Support the termination of PBN spanning trees by inhibiting transmission of PBN BPDUs at a Provider Instance Port.

5.7.1 I-component options

A conformant I-component may implement any of the options specified for an S-VLAN component (5.6.1), and may

- a) Support many-to-one mapping from S-VID values to I-SID values;
- b) Support a common spanning tree among PBNs interconnected with one or more Backbone service instances by permitting transmission of PBN BPDUs through Virtual Instance Ports at a Provider Instance Port;
- c) Filter frames received at Provider Instance Ports that have the B-SA used by the Provider Instance Port, to prevent such frames from circulating if Backbone service instances in two or more attached PBBNs are configured in a loop;

- d) Use back-to-back Backbone service instance multiplex entities (6.18) on a PIP to enable monitoring of Backbone service instances with Connectivity Fault Management (CFM).

5.8 B-component conformance

A B-component comprises an S-VLAN component with the EISS on each Provider Network Port supported by the use of a Service VLAN tag (6.9, 9.5), and the EISS on each Customer Backbone Port supported by the use of both a Service VLAN tag (6.9, 9.5) and a Backbone Service Instance tag (6.11, 9.5). A conformant implementation of a B-component shall

- a) Comprise a single conformant S-VLAN component;
- b) Recognize and use Backbone Service Instance tags (I-TAGs) on one or more Customer Backbone Ports (6.11);
- c) Terminate PBBN spanning tree by inhibiting transmission of PBBN BPDUs at a Customer Backbone Port.

5.8.1 B-component options

A conformant B-component may implement any of the options specified for an S-VLAN component (5.6.1), and may

- a) Translate I-SID values by supporting the Local-SID field in the Backbone Service Instance table in Customer Backbone Ports (6.11);
- b) Assign B-VID values based on I-SID values by supporting the B-VID field in the Backbone Service Instance table in Customer Backbone Ports (6.11);
- c) Use the Default Backbone Destination field in the Backbone Service Instance table in Customer Backbone Ports (6.11) as the destination MAC address for any frames received with the Backbone Service Instance group address (26.4);
- d) Use back-to-back Backbone Service Instance multiplex entities (6.18) on a CBP to enable monitoring of Backbone service instances with CFM.

5.8.2 B-component requirements for PBB-TE

A B-component implementation that conforms to the provisions of this standard for PBB-TE (25.10) shall

- a) Disable learning for a set of B-VIDs allocated to TE-MSTID as specified in 8.4 and in 8.9
- b) Discard frames with unregistered destination addresses for B-VIDs allocated to TE-MSTID (8.8.1)
- c) Support the Default Backbone Destination field in the Backbone service instance tables on the CBPs providing TESIs as specified in 6.11
- d) Support the Backbone VLAN identifier field in the Backbone service instance tables on the CBPs providing TESIs as specified in 6.11
- e) Support the creation of MEPs for the TESIs on CBPs as specified in 26.9
- f) Support 1:1 protection switching as specified in 26.10.3
- g) Provide control via all of the PBB-TE required managed objects specified in Clause 12

A B-component implementation that conforms to the provisions of this standard for PBB-TE (25.10) may

- h) Allow control of all B-VIDs by an external agent, conforming to the Reserved VID values of Table 9-2
- i) Support MIP creation and the corresponding CFM extensions for TESIs as specified in 26.9
- j) Support the Hold-Off timer used by the protection switching protocol as specified in 26.10.3

- k) Support sharing of TESIs among TE protection groups in order to provide 1:1 protection switching that is capable load sharing as specified in 12.14.1.2 and 26.10.2
- l) Support the Mismatch defect identification as specified in 26.11
- m) Support the PBB-TE Management Information Base (MIB) module defined in Clause 17

5.9 VLAN Bridge conformance

A VLAN Bridge shall comprise a single conformant C-VLAN component (5.5).

Each VLAN Bridge Port shall be capable of attaching directly to an IEEE 802 LAN.

5.9.1 VLAN Bridge options

A VLAN Bridge may implement any C-VLAN component option (5.5).

5.10 Provider Bridge conformance

A Provider Bridge shall comprise a single conformant S-VLAN (5.6) component and zero or more C-VLAN components (5.5).

Each Port shall be capable of being configured as one of, and may be capable of being configured as any of

- a) A Provider Network Port;
- b) A Customer Network Port;
- c) A Customer Edge Port;

as specified in Clause 15. Each Port configured as a Provider Network Port or Customer Network Port shall be capable of attaching the S-VLAN component of the Provider Bridge directly to an IEEE 802 LAN. Each Port configured as a Customer Edge Port shall be capable of attaching a C-VLAN component within the Provider Bridge directly to an IEEE 802 LAN.

5.10.1 S-VLAN Bridge conformance

An S-VLAN Bridge shall comprise a single conformant S-VLAN component (5.6). An S-VLAN Bridge does not have any physical interfaces configured as a Customer Edge Port, nor does it include any C-VLAN components.

5.10.2 Provider Edge Bridge conformance

A Provider Edge Bridge is a conformant Provider Bridge with the capability to include one or more C-VLAN components as specified in 15.4.

Each C-VLAN component shall comprise a single Customer Edge Port and a single distinct Provider Edge Port for each service instance that can be provided through that Customer Edge Port. Each Provider Edge Port shall be connected within the Provider Edge Bridge, as specified in 6.14, to a distinct Customer Network Port on the S-VLAN component. Each C-VLAN component shall implement RSTP, with the enhancements to support Customer Edge Ports, as specified in 13.39.

NOTE—The single Customer Edge Port supported by a C-VLAN component can be supported by two or more independent instances of a MAC, aggregated as specified by Link Aggregation (IEEE Std 802.1AX™).

5.11 Backbone Edge Bridge conformance

A Backbone Edge Bridge system shall comprise zero or more I-components and zero or one B-component and zero or more T-components, but at least one I-component or one B-component or one T-component, supporting externally accessible ports as specified in Clause 25. Each externally accessible port shall be designated as one of, and may be capable of being configured as any of the following:

- a) A Provider Network Port
- b) A Provider Instance Port
- c) A Customer Backbone Port
- d) A Customer Network Port
- e) A Customer Edge Port

5.11.1 Backbone Edge Bridge requirements for PBB-TE

A Backbone Edge Bridge that conforms to the provisions of this standard for PBB-TE (25.10) shall comprise one B-component capable of providing TESIs (5.8.2) and one or more I-components (5.7) or T-components (5.15).

5.12 VLAN-unaware Bridge component requirements

An implementation of a VLAN-unaware Bridge component shall

- a) Conform to the requirements of IEEE Std 802.1D, as modified by the provisions of this standard.
- b) Conform to the relevant standard for the Media Access Control technology implemented at each Port in support of the MAC ISS, as specified in 6.6 and 6.7;
- c) Support the MAC Relay Entity by means of an ISS interface (not an EISS interface).

5.13 TPMR component conformance

A TPMR component comprises a VLAN-unaware Bridge component. A conformant implementation of a TPMR component shall

- a) Support exactly two Bridge Ports other than a Management Port.
- b) Support the operation of the following functions of the Forwarding Process:
 - 3) Frame filtering (8.6.3)
 - 4) Queuing frames (8.6.6)
 - 5) Transmission selection (8.6.8)
- c) Support the operation of the MAC status propagation entity, by means of the functions and protocol specified in Clause 23.
- d) Support at least one traffic class on each Bridge Port.
- e) Support the management functionality specified in 12.19.
- f) Support only the following entries in the Filtering Database:
 - 1) Static filtering entries, permanently configured in the Filtering Database and that cannot be modified by management, that specify filtering on both of the TPMR's Bridge Ports for the group MAC addresses defined in Table 8-3; and
 - 2) A static filtering entry, permanently configured in the Filtering Database and that cannot be modified by management, that specifies forwarding or filtering on each of the TPMR's Bridge Ports for the MAC address associated with the management entity of the TPMR. The values that are configured into this filtering entry are determined by the implementer, and shall be stated in the PICS.

- g) Support a MIP on a MA that is not associated with any VLAN on each of the TPMR's Bridge Ports.
The default MD level shall be zero.

NOTE—As a TPMR is a non-VLAN aware Bridge, these maintenance points are not associated with a particular VID (see 22.1.7). This default level 0 MIP allows detection and identification of the TPMR without configuration. Such a MIP can of course be disabled by management.

A conformant implementation of a TPMR component shall not

- h) Implement the Rapid Spanning Tree Protocol defined in Clause 13.
i) Encode transmitted BPDUs and validate received BPDUs (Clause 14).
j) Support Extended Filtering Services for relaying and filtering frames (6.16.5).

5.13.1 TPMR component options

An implementation of a TPMR component may

- a) Support multiple traffic classes on each Bridge Port.
b) Support the ISS with signaled priority on each Bridge Port (6.20).

5.14 TPMR conformance

An implementation of a TPMR shall

- a) Comprise a single conformant TPMR component (5.13).
b) Support exactly two externally accessible Ports, each capable of attaching directly to an IEEE 802 LAN.
c) Be capable of supporting a management entity using the TPMR Port Connectivity (8.5.2) on one of its externally accessible ports.
d) Support remote management of the TPMR managed objects specified in 12.19.

5.14.1 TPMR options

A TPMR may implement any TPMR component option (5.13.1).

An implementation of a TPMR may

- a) Support a management entity using the Bridge Port Connectivity (8.5.1) on a Management Port (8.3) or the externally accessible ports.
b) Support remote management using IETF RFC 4789, Simple Network Management Protocol (SNMP) over IEEE 802 Networks (see 8.13.7).
c) Support remote management using the TPMR MIB module defined in 17.7.11.

NOTE—While IETF RFC 4789 is specifically referenced, the standard does not preclude the support of other transports for SNMP.

5.15 T-component conformance

A T-component comprises a TPMR component (5.13) with one Provider Instance Port (6.11).

A conformant implementation of an T-component shall

- a) Comprise a single conformant TPMR component.

- b) Recognize and use Backbone Service Instance tags (I-TAGs) on one Provider Instance Port (6.10).
- c) Support a single Virtual Instance Port at the Provider Instance Port.

5.15.1 T-component options

A conformant T-component may implement any of the options specified for a TPMR component (5.13.1).

5.16 End station requirements for MMRP, MVRP, and MSRP

This subclause defines the conformance requirements for end station implementations claiming conformance to MVRP, MMRP, and MSRP. Although this standard is principally concerned with defining the requirements for VLAN-aware Bridges, the conformance requirements for end station implementations of MVRP, MMRP, and MSRP are included in order to give guidance to such implementations. The PICS Proforma defined in Annex I is concerned with conformance claims with respect to end station implementations.

For the reasons stated in 10.6, it is recommended that end stations that do not require the ability to perform Source Pruning implement the Applicant-Only Participant, in preference to the Simple-Applicant Participant.

5.16.1 MMRP requirements and options

An end station for which conformance to MMRP is claimed shall

- a) Conform to the operation of the MRP Applicant, Registrar, LeaveAll, and Periodic Transmission state machines, as defined in 10.7.7, 10.7.8, 10.7.9, and 10.7.10, as required for one or more of the following variants of the MRP Participant, as defined in 10.6 and 10.7:
 - 1) The Full Participant.
 - 2) The Full Participant, point-to-point subset.
 - 3) The Applicant-Only Participant.
 - 4) The Simple-Applicant Participant.
- b) Exchange MPDUs as required by the MRP state machine(s) implemented, formatted in accordance with the generic PDU format described in 10.8, and able to carry application-specific information as defined in 10.12.1, using the MMRP Application address as defined in Table 10-1.

An end station for which conformance to the operation of the Applicant state machine (10.7.7) is claimed may

- c) Perform source pruning, as defined in 10.10.3 and 10.12.

It is recommended that only those end stations that require the ability to perform Source Pruning (10.10.3) conform to the operation of the Full Participant or the Full Participant, point-to-point subset.

NOTE—If an end station does not require the ability to perform Source Pruning, then there is no need for it to implement Registrar functionality.

5.16.2 MVRP requirements and options

An end station for which conformance to MVRP is claimed shall

- a) Conform to the operation of the MRP Applicant, Registrar, LeaveAll, and Periodic Transmission state machines, as defined in 10.7.7, 10.7.8, 10.7.9, and 10.7.10, as required for one or more of the following variants of the MRP Participant, as defined in 10.6 and 10.7:

- 1) The Full Participant.
 - 2) The Full Participant, point-to-point subset.
 - 3) The Applicant-Only Participant.
 - 4) The Simple-Applicant Participant.
- b) Exchange MPDUs as required by the MRP state machine(s) implemented, formatted in accordance with the generic PDU format described in 10.8, and able to carry application-specific information as defined in 11.2.3, using the MVRP Application address as defined in Table 10-1.

An end station for which conformance to MVRP is claimed may

- c) Perform source pruning, as defined in 10.10.3 and 11.2.1.1.

It is recommended that only those end stations that require the ability to perform Source Pruning (11.2.1.1) conform to the operation of the Full Participant or the Full Participant, point-to-point subset.

NOTE—If an end station does not require the ability to perform Source Pruning, then there is no need for it to implement Registrar functionality.

5.16.3 MSRP requirements and options

An end station for which conformance to MSRP is claimed shall:

- a) Conform to the operation of the MRP Applicant, Registrar, and LeaveAll state machines, as defined in 10.7.7, 10.7.8, and 10.7.9, as required for one or more of the following variants of the MRP Participant, as defined in 10.6 and 10.7:
 - 1) The Full Participant.
 - 2) The Full Participant, point-to-point subset.
 - 3) The Applicant-Only Participant.
 - 4) The Simple-Applicant Participant.
- b) Exchange MPDUs as required by the MRP state machine(s) implemented, formatted in accordance with the generic PDU format described in 10.8, and able to carry application-specific information as defined in 35.2.2, using the MSRP Application address as defined in Table 10-1.
- c) Not redeclare its attributes unless responding to a LeaveAll event.

An end station for which conformance to the operation of the Applicant state machine (10.7.7) is claimed may

- d) Perform Talker pruning, as defined in 35.2.1.4(b), 35.2.3.1 and 35.2.4.3.1.
- e) Perform Listener pruning as described in 35.2.3.1.

NOTE—In order to meet the requirement specified in 5.16.3(c), the Periodic Transmission state machine (10.7.10) is specifically excluded from MSRP. An end station is allowed to generate an MSRP LeaveAll event to force an immediate redeclaration of all MSRP attributes from its neighbor(s).

5.17 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 through Clause 22) shall:

- a) Support the creation of Maintenance Domains at eight MD Levels, with multiple nonoverlapping 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 VID supported by the Station for each MD Level;

- c) Support the creation of eight Down MEPs on each VID on each Port, each MEP at a different MD Level;
- d) Support the creation of eight Down MEPs attached to no VID 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 VID (see 22.1.5).

5.18 End station requirements—forwarding and queuing for time-sensitive streams

An end station implementation that conforms to the provisions of this standard for forwarding and queuing for time-sensitive streams shall:

- a) Support a minimum of two traffic classes on all Ports, of which
 - 1) A minimum of one traffic class supports the strict priority algorithm for transmission selection (8.6.8.1), and
 - 2) One traffic class is an SR class.
- b) Support the operation of the credit-based shaper algorithm (8.6.8.2) as the transmission selection algorithm used for frames transmitted for each stream associated with the SR class.
- c) Support the operation of the credit-based shaper algorithm (8.6.8.2) on all Ports as the transmission selection algorithm used for the SR class.
- d) Use the default priority associated with SR class “B” as shown in Table 6-6 as the priority value carried in transmitted SR class “B” data frames.

An end station implementation that conforms to the provisions of this standard for forwarding and queuing for time-sensitive streams may:

- e) Support two or more SR classes (a maximum of seven), and support the operation of the credit-based shaper algorithm (8.6.8.2) on all Ports as the transmission selection algorithm used for those SR classes. The number of SR classes supported shall be stated in the PICS.
- f) Use the default priority associated with SR class “A” as shown in Table 6-6 as the priority value carried in transmitted SR class “A” data frames. If more than two SR classes are supported, the priority value carried in transmitted data frames for the additional SR classes shall be stated in the PICS.

5.19 End station requirements for congestion notification

A VLAN-aware end station implementation that conforms to the provisions of this standard for congestion notification (Clause 30 through Clause 33) shall

- a) Support the creation of at least one Reaction Point (31.2.2.2);
- b) Support, if Congestion Points are supported, the ability to configure the variables controlling the operation of each Congestion Point, if any (12.21.1, 12.21.2, 12.21.3, 12.21.4);
- c) Support, at each supported Congestion Point, if any, the generation of Congestion Notification Messages (32.7);

- d) Support the ability to configure the variables controlling the operation of each Reaction Point (12.21.1, 12.21.5, 12.21.6);
- e) Support the operation of its Port in at least the cptDisabled and cptInterior Congestion Notification Domain defense modes separately for each Congestion Notification Priority Value (32.1.1);
- f) Conform to the required specifications of the Link Layer Discovery Protocol (LLDP) of IEEE Std 802.1AB;
- g) Support the use of the Congestion Notification TLV in LLDP (32.1.1);
- h) Support, in each Reaction Point, the limiting of frames output in response to the reception of Congestion Notification Messages (31.2.2.2);
- i) Support, if multiple Reaction Points are supported, the ability to distinguish among Congestion Notification Messages pertaining to different Reaction Points, and direct each such message to the correct Reaction Point (31.2.5).

A VLAN-aware end station implementation that conforms to the provisions of this standard for congestion notification (Clause 30 through Clause 33) may

- j) Support the creation of one or more Congestion Points (31.1.1);
- k) Support the creation of more than one Reaction Point (31.2.2.2);
- l) Support more than one Congestion Notification Priority Value per Reaction Point (31.2.1); and/or
- m) Support the operation of its Port in both the cptInterior and cptInteriorReady Congestion Notification Domain defense modes separately for each Congestion Notification Priority Value (31.1.1);
- n) Support the IEEE8021-CN-MIB (17.7.13).

5.20 MAC-specific bridging methods

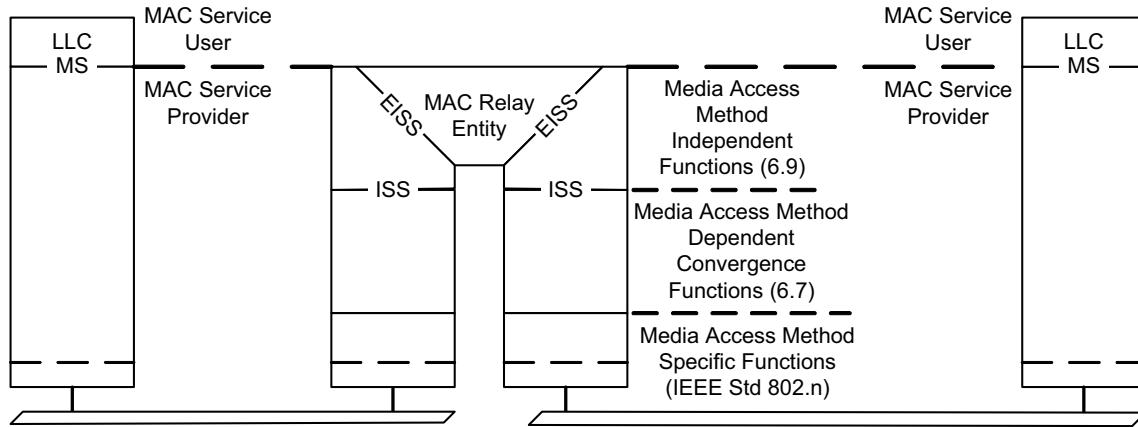
MAC-specific bridging methods may exist. Use of a MAC-specific bridging method and the method specified in this standard on the same LAN shall

- a) Not prevent communication between stations in a network.
- b) Preserve the MAC Service.
- c) Preserve the characteristics of each bridging method within its own domain.
- d) Provide for the ability of both bridging techniques to coexist simultaneously on a LAN without adverse interaction.

6. Support of the MAC Service

VLAN-aware MAC Bridges interconnect the separate IEEE 802 LANs that compose a Virtual Bridged Local Area Network by relaying and filtering frames between the separate MACs of the bridged LANs.

The position of a VLAN-aware Bridge's MAC Relay Entity (8.2) within the MAC Sublayer is shown in Figure 6-1.



NOTE—The notation “IEEE Std 802.n” in this figure indicates that the specifications for these functions can be found in the relevant standard for the media access method concerned; for example, n would be 3 (IEEE Std 802.3) in the case of Ethernet.

Figure 6-1—Internal organization of the MAC sublayer

The MAC Sublayer comprises:

- a) Media access method specific functions¹⁸ that realize transmission and reception of MAC Protocol Data Units (MPDUs);
- b) Media access method dependent convergence functions that use item a) to provide a media access method independent service;
- c) Media access method independent functions that use a media independent service to provide the same or another media independent service.

A VLAN-unaware Bridge's MAC Relay Entity forwards frames between the instances of the media independent Internal Sublayer Service (6.6).

A VLAN-aware Bridge's MAC Relay Entity forwards frames between the instances of the media independent Enhanced Internal Sublayer Service (EISS, 6.8). The EISS is provided by the functions specified in 6.9 using the media independent Internal Sublayer Service (6.6). The convergence functions that provide the ISS using the media specific functions for each IEEE 802 LAN MAC type are specified in 6.7.

This clause:

- d) 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);
- e) Describes how Bridges preserve and maintain the quality of the MAC Service (6.4, 6.5);

¹⁸The media access method specific functions together with media access method dependent convergence functions that realize a MAC Service for use in end stations are specified for each IEEE 802 LAN media access control method or “MAC type” (e.g., IEEE 802.3, IEEE 802.11) by the relevant standard for that media access control method and are commonly referred to as “the MAC.”

- f) Specifies the MAC Internal Sublayer Service (ISS) and MAC status parameters, their support by specific media access methods (6.6, 6.7), and support for signaled priority (6.6, 6.7, 6.14, 6.15, 6.20);
- g) Specifies the Enhanced Internal Sublayer Service (EISS) used by VLAN-aware stations, the encoding of VID and priority parameters in transmitted frames, and the classification of received frames that do not explicitly convey those parameters (6.8, 6.9, 6.12); and
- h) Specifies how entities defined in terms of the ISS can support the EISS (6.17).

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.¹⁹

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 are 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 6-2 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 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.²⁰

¹⁹Drawing on prior network layer standards, including ISO/IEC 7498-1, wherever possible.

²⁰These 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.

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.

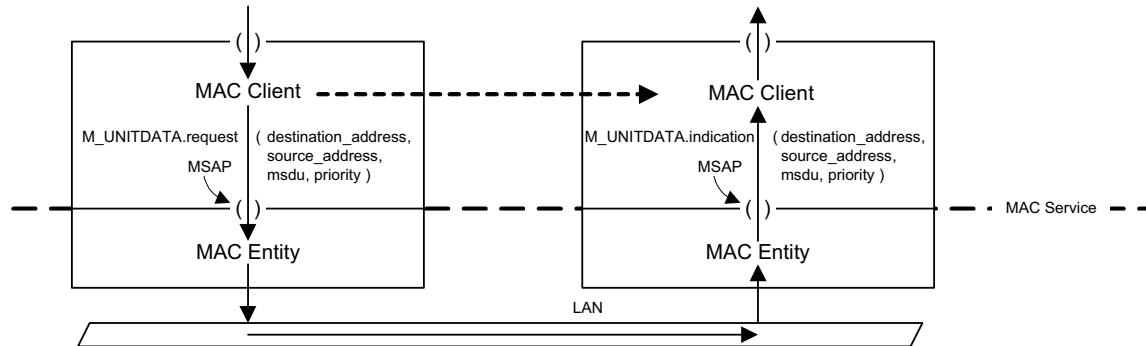


Figure 6-2—MAC entities, the MAC Service, and MAC Service users (clients)

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 primitives²¹ 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.

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). Each service access point has an identifier with a value that is local to the system and uniquely identifies the service access point within the system.

²¹This 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.

The network and link layers²² 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 6-3 provides an example, showing the use of Link Aggregation (IEEE Std 802.1AX).

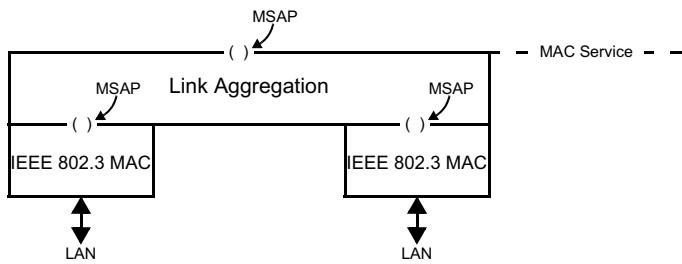


Figure 6-3—An interface stack

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 6-3, 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 [B10], 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 802.1AX), security (e.g., IEEE Std 802.1AE™-2006 [B7]), or multiplexing.

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. Within a Bridge, the MAC Relay Entity is a client of the ISS or EISS, and the LLC Entity is a client of the MAC Service. The LLC Entity is specified in ISO/IEC 8802.2 and 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 Length/Type field as specified in IEEE Std 802a™-2003 [B6].

²²The 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 (ISO/IEC 8802-2) and HDLC was sought, together with the adoption of a unique network layer protocol (ITU-T X.25 (1996) [B41]). 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.

6.1.7 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.²³

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).

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

²³A LAN is thus defined in terms of its external observable behavior, not by an abstraction of its internal operation.

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.

6.2.1 Point-to-point, multipoint-to-multipoint, and rooted-multipoint connectivity

The types of connectivity supported by associations of Bridge Ports in Virtual Bridged LANs are often classified into three categories: point-to-point, multipoint-to-multipoint, and rooted-multipoint. Point-to-point connectivity is an association of exactly two Bridge Ports such that an ingress frame (for a given VLAN) at one Bridge Port may result in an egress frame at the other Bridge Port. Multipoint-to-multipoint connectivity is an association of two or more Bridge Ports such that an ingress frame at one Bridge Port may result in an egress frame at any or all of the other Bridge Ports. Rooted-multipoint connectivity is an association of two or more Bridge Ports where each Bridge Port is designated as either a Root or a Leaf Port, and an ingress frame at a Root Port may result in an egress frame at any or all Leaf Ports and other Root Ports, while an ingress frame at a Leaf Port may result in an egress frame at any or all Root Ports but never results in an egress frame at another Leaf Port. Note that point-to-point connectivity is a special case of the more general multipoint-to-multipoint connectivity, and that both point-to-point connectivity and multipoint-to-multipoint connectivity are special cases of rooted-multipoint connectivity where all Bridge Ports are designated as Root Ports.

6.3 Support of the MAC service

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 VID(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 VID. 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 VID 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 VID 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 can use the EISS Multiplex Entity (6.17) to provide multiple SAPs, one per VID of interest, to separate MAC Clients.

The objective of Bridge operation is to maximize the availability of the MAC Service to end stations and to assist in the maintenance of the network. Bridges can be configured to provide redundant paths between end stations, enabling the network to continue providing service in the event of component failure (of Bridge or LAN). The paths supported between end stations are predictable and configurable. Group addressed (multicast and broadcast) frames for any given VLAN take the same path as frames addressed to an individual end station (unicast), thus permitting the use of either individually addressed or group addressed

frames to configure or diagnose paths for frames with either group or individual destination addresses. The paths taken for frames for any given VLAN between any two stations are symmetric, i.e. frames from station A (say) to station B traverse the same bridges and LANs as those from B to A, only in the reverse order. This symmetry reduces the probability of one-way connectivity between stations and permits the use of station location learning to localize traffic (6.5.10).

The VLAN identifier (VID) conveyed in the VLAN tag associates each frame with the following:

- a) A VLAN.
- b) A loop-free active topology.

6.4 Preservation of the MAC service

The MAC Service offered by a network consisting of LANs interconnected by Bridges is similar to that offered by a single LAN (see 6.5).

- a) Frames transmitted between end stations carry the MAC Addresses of the peer-end stations in their destination and source address fields, not an address of a Bridge. The MAC Relay is not directly addressed by communicating end stations.
- b) The MAC Addresses of end stations are not restricted by the network's topology or configuration.
- c) All MAC Addresses need to be unique within a VLAN, and within any set of VLANs for which filtering information is shared by a Bridge.

6.5 Quality of service maintenance

6.5.1 Service availability

Service availability is measured as that fraction of some total time during which the MAC Service is provided. The operation of a Bridge can increase or lower the service availability.

The service availability can be increased by automatic reconfiguration of the network in order to avoid the use of a failed component (e.g., repeater, cable, or connector) in the data path. The service availability can be lowered by failure of a Bridge, through denial of service by the Bridge, or through frame filtering by the Bridge. Changes in topology, caused by component failures, the addition or removal of components, or administrative changes, are detected and signaled by the following means:

- a) Physical detection of component failure and signaling of that failure by the Enhanced Internal Sublayer Service (6.8 and 6.9);
- b) Detection of component failure through the operation of a spanning tree algorithm and protocol;
- c) Explicit signaling of reconfiguration events through the operation of a spanning tree algorithm and protocol.

Automatic reconfiguration can be achieved rapidly on the detection of a physical topology change (see Clause 13), thus minimizing any service denial that is caused by the reconfiguration.

A Bridge may deny service and discard frames (6.5.2) in order to preserve other aspects of the MAC Service (6.5.3 and 6.5.4) when automatic reconfiguration takes place. Service may be denied to end stations that do not benefit from the reconfiguration; hence, the service availability is lowered for those end stations. Bridges may filter frames in order to localize traffic in the network. Should an end station move, it may then be unable to receive frames from other end stations until the filtering information held by the Bridges is updated.

To minimize the effects of service denial caused by reconfiguration events, filtering information that has been dynamically learned can be modified when automatic reconfiguration takes place, or in preparation for future reconfiguration events (Clause 13 and 13.15). However, filtering information that is statically configured cannot be modified in this way.

A Bridge may deny service and discard frames in order to prevent access to the network by devices that are not authorized for such access.

To maximize the service availability, no loss of service or delay in service provision should be caused by Bridges, except as a consequence of a failure, removal, or insertion of a network component; or as a consequence of the movement of an end station; or as a consequence of an attempt to perform unauthorized access. These events are regarded as extraordinary. The operation of any additional protocol necessary to maintain the quality of the MAC Service is thus limited to the configuration of the network and is independent of individual instances of service provision.

NOTE 1—This is true only in circumstances where admission control mechanisms are not present, i.e., where the Bridges provide a “best effort” service.

NOTE 2—The operation of management on the Bridge can result in the Bridge being reset, either as a result of a specific Bridge reset operation or as a consequence of manipulating the Bridge’s configuration. From the point of view of service availability, resetting the Bridge is an extraordinary event that has a similar effect to physical removal of the Bridge from the network, followed by reinsertion of the Bridge into the network.

6.5.2 Frame loss

The MAC Service does not guarantee the delivery of Service Data Units. Frames transmitted by a source station arrive, uncorrupted, at the destination station with high probability. The operation of a Bridge introduces minimal additional frame loss.

A frame transmitted by a source station can fail to reach its destination station as a result of

- a) Frame corruption during physical layer transmission or reception.
- b) Frame discard by a Bridge because
 - 1) It is unable to transmit the frame within some maximum period of time and, hence, must discard the frame to prevent the maximum frame lifetime (6.5.6) from being exceeded.
 - 2) It is unable to continue to store the frame due to exhaustion of internal buffering capacity as frames continue to arrive at a rate in excess of that at which they can be transmitted.
 - 3) The size of the service data unit carried by the frame exceeds the maximum supported by the MAC procedures employed on the LAN to which the frame is to be relayed.
 - 4) Changes in the connected topology of the network necessitate frame discard for a limited period of time to maintain other aspects of Quality of Service (13.15).
 - 5) The device attached to the Port is not authorized for access to the network.
 - 6) The configuration of Static Filtering Entries or Static VLAN Registration Entries in the Filtering Database (8.8.1, 8.8.2) disallows the forwarding of frames with particular destination addresses or VLAN classifications on specific Ports.
 - 7) A flow metering algorithm (8.6.5) determines that discard is necessary.

NOTE—As Static Filtering Entries and Static VLAN Registration Entries are associated with particular Ports or combinations of Ports, there is a possibility that misconfiguration of such entries will lead to unintended frame discard during or following automatic reconfiguration of the network.

6.5.3 Frame misordering

The MAC Service (9.2 of ISO/IEC 15802-1) permits a negligible rate of reordering of frames with a given priority for a given combination of destination address and source address, transmitted on a given VLAN.

MA_UNITDATA.indication service primitives corresponding to MA_UNITDATA.request primitives, with the same requested priority and for the same combination of VLAN classification, destination address, and source address, are received in the same order as the request primitives were processed.

NOTE 1—The operation of the Forwarding Process in Bridges (8.6) is such that the frame-ordering characteristics of the MAC Service are preserved.

Where Bridges in a network are capable of connecting the individual MACs in such a way that multiple paths between any source station–destination station pairs exist, the operation of a protocol is required to ensure that a single path is used.

NOTE 2—Where STP is in use (see Clause 8 of IEEE Std 802.1D, 1998 Edition), frame misordering cannot occur during normal operation. Where RSTP is in use (see Clause 13), there is an increased probability that frames that are in transit through the network will be misordered, because a Bridge can buffer frames awaiting transmission through its Ports. The probability of misordering occurring as a result of such an event is dependent on implementation choices and is associated with Spanning Tree reconfiguration events. Some known LAN protocols, for example, LLC Type 2, are sensitive to frame misordering and duplication; in order to allow Bridges that support RSTP to be used in environments where sensitive protocols are in use, the forceVersion parameter (13.6.2) can be used to force a Bridge that supports RSTP to operate in an STP-compatible manner. A more detailed discussion of misordering in RSTP can be found in K.3 of IEEE Std 802.1D-2004.

6.5.4 Frame duplication

The MAC Service (9.2 of ISO/IEC 15802-1) permits a negligible rate of duplication of frames. The operation of Bridges introduces a negligible rate of duplication of user data frames.

The potential for frame duplication in a network arises through the possibility of duplication of received frames on subsequent transmission within a Bridge, or through the possibility of multiple paths between source and destination end stations.

Where Bridges in a network are capable of connecting the individual MACs in such a way that multiple paths between any source station–destination station pairs exist, the operation of a protocol is required to ensure that a single path is used.

NOTE—Where RSTP is in use (see Clause 13), there is an increased probability that a Spanning Tree reconfiguration event can cause frames that are in transit through the network to be duplicated, because a Bridge can buffer frames awaiting transmission through its Ports. As the probability of duplication occurring as a result of such an event is small, and the frequency of Spanning Tree reconfiguration events is also small, the degradation of the properties of the MAC service caused by this source of frame duplication is considered to be negligible. A more detailed discussion of frame duplication in RSTP can be found in K.2 of IEEE Std 802.1D-2004.

6.5.5 Transit delay

The MAC Service introduces a frame transit delay that is dependent on the particular media and MAC method employed. Frame transit delay is the elapsed time between an MA_UNITDATA.request primitive and the corresponding MA_UNITDATA.indication primitive. Elapsed time values are calculated only on Service Data Units that are successfully transferred.

Since the MAC Service is provided at an abstract interface within an end station, it is not possible to specify precisely the total frame transit delay. It is, however, possible to measure those components of delay associated with media access and with transmission and reception; and the transit delay introduced by an intermediate system, in this case a Bridge, can be measured.

The minimum additional transit delay introduced by a Bridge is the time taken to receive a frame plus that taken to access the media onto which the frame is to be relayed. Note that the frame is completely received before it is relayed as the Frame Check Sequence (FCS) is to be calculated and the frame discarded if in error.

6.5.6 Frame lifetime

The MAC Service ensures that an upper bound to the transit delay is experienced for a particular instance of communication. This maximum frame lifetime is necessary to ensure the correct operation of higher layer protocols. The additional transit delay introduced by a Bridge is discussed in 6.5.5.

To enforce the maximum frame lifetime, a Bridge may be required to discard frames. Since the information provided by the MAC Sublayer to a Bridge does not include the transit delay already experienced by any particular frame, Bridges discard frames to enforce a maximum delay in each Bridge.

The value of the maximum bridge transit delay is based on both the maximum delays imposed by all Bridges in the network and the desired maximum frame lifetime. A recommended and an absolute maximum value are specified in Table 7-3 of IEEE Std 802.1D-2004.

6.5.7 Undetected frame error rate

The MAC Service introduces a very low undetected frame error rate in transmitted frames. Undetected errors are protected against by the use of an FCS that is appended to the frame by the MAC Sublayer of the source station prior to transmission, and checked by the destination station on reception.

The FCS calculated for a given service data unit is dependent on the MAC method employed. It is therefore necessary to recalculate the FCS within a Bridge providing a relay function between IEEE 802 LAN MACs of dissimilar types if differences in the method of calculation and/or the coverage of the FCS, or changes to the data that is within the coverage of the FCS, would lead to a different FCS being calculated for the service data unit by the two MAC methods. This introduces the possibility of additional undetected errors arising from the operation of a Bridge. For frames relayed between LANs of the same MAC type, the Bridge shall not introduce an undetected frame error rate greater than that which would have been achieved by preserving the FCS.

NOTE—Application of the techniques described in Annex F of IEEE Std 802.1D-2004 allows an implementation to achieve an arbitrarily small increase in undetected frame error rate, even in cases where the data that is within the coverage of the FCS is changed. As a maintenance activity on this standard, revision of the wording of this requirement will be initiated, with a view to placing a quantitative limit on the increase in undetected frame error rate that is acceptable in a conformant implementation.

6.5.8 Maximum Service Data Unit Size

The Maximum Service Data Unit Size that can be supported by an IEEE 802 LAN varies with the MAC method and its associated parameters (e.g., speed, electrical characteristics). It may be constrained by the owner of the LAN. The Maximum Service Data Unit Size supported by a Bridge between two LANs is the smaller of that supported by the LANs. No attempt is made by a Bridge to relay a frame to a LAN that does not support the size of Service Data Unit conveyed by that frame.

6.5.9 Priority

The MAC Service includes priority as a Quality of Service parameter. MA_UNITDATA.request primitives with a high priority may be given precedence over other request primitives made at the same station, or at other stations attached to the same LAN and can give rise to earlier MA_UNITDATA.indication primitives.

The MAC Sublayer maps the requested priority onto the priorities supported by the individual MAC method. The requested priority may be conveyed to the destination station.

The transmission delay experienced by a frame in a Bridge can be managed by associating a priority with the frame.

The transmission delay comprises

- a) A queuing delay until the frame becomes first in line for transmission on the Port, in accordance with the procedure for selecting frames for transmission described in 8.6.8;
- b) The access delay for transmission of the frame.

Queuing delays can be managed using priority. Access delays can be managed using priority in MAC methods that support more than one priority.

The priority associated with a frame can be signaled by means of the priority signaling mechanisms inherent in some IEEE 802 LAN MAC types. Since not all IEEE 802 LAN MAC types are able to signal the priority associated with a frame, VLAN-aware Bridges regenerate priority based on a combination of signaled information and configuration information held in the Bridge.

The Bridge maps priority values onto one or more traffic classes; Bridges that support more than one traffic class are able to support expedited classes of traffic. The Forwarding Process, 8.6, describes the use of priority and traffic classes in Bridges. Given the constraints placed on frame misordering in a Bridge, as expressed in 6.5.3, the mappings of priority and traffic class are static.

NOTE 1—The term “traffic class,” as used in this standard, is used only in the context of the operation of the priority handling and queueing functions of the Forwarding Process, as described in 8.6. Any other meanings attached to this term in other contexts do not apply to the use of the term in this standard.

The ability to signal priority in IEEE 802 LANs, coupled with a consistent approach to the mapping of priority to traffic classes, and of priority to the priority requested from the individual LAN or supporting service, allows consistent use of priority information to be made, according to the capabilities of the Bridges and MAC methods that are involved in the transmission path.

NOTE 2—This standard defines a frame format and associated procedures that can be used to carry priority information across LAN MAC types that are not able to signal priority.

Under normal circumstances, priority is not modified in transit through the relay function of a Bridge; however, there may be some circumstances where it is desirable for management purposes to control how priority is propagated. The Priority Regeneration Table (Table 6-5) provides the ability to map incoming priority values on a per-Port basis, under management control. In its default state, this table provides an identity mapping from priority values to Regenerated priority values; i.e., by default, the Regenerated priority is identical to the incoming priority.

6.5.10 Throughput

The total throughput provided by a network can be significantly greater than that provided by an equivalent single LAN. Bridges may localize traffic within the network by filtering frames. Filtering services available in bridged networks are described in 6.16.

The throughput between end stations on individual LANs, communicating through a Bridge, can be lowered by frame discard in the Bridge due to the inability to transmit at the required rate on the LAN forming the path to the destination for an extended period.

6.6 Internal Sublayer Service

The Internal Sublayer Service (ISS) augments the specification of the MAC Service (ISO/IEC 15802-1) with elements necessary to the performance of the relay function. Within an end station, these additional elements are considered either to be below the MAC Service boundary, and pertinent only to the operation of the service provider, or to be local matters not forming part of the peer-to-peer nature of the MAC

Service. The ISS excludes MAC-specific features and procedures whose operation is confined to an individual LAN.

NOTE—No new service primitives are defined. The frame_check_sequence, drop_eligible, service_access_point_identifier, and connection_identifier are added to the list of parameters associated with the MA_UNITDATA.request and MA_UNITDATA.indication primitives.

6.6.1 Service primitives and parameters

The ISS is specified by two unit-data primitives, an M_UNITDATA.indication and an M_UNITDATA.request, together with the parameters of those primitives. Each M_UNITDATA indication corresponds to the receipt of an error-free MAC frame from a LAN. A data request primitive is invoked to transmit a frame to an individual LAN.

NOTE 1—Detailed specifications of error conditions in received frames are contained in the relevant MAC standards; for example, FCS errors, length errors, and nonintegral number of octets.

```
M_UNITDATA.indication  (
    destination_address,
    source_address,
    mac_service_data_unit,
    priority,
    drop_eligible,
    frame_check_sequence,
    service_access_point_identifier,
    connection_identifier
)

M_UNITDATA.request      (
    destination_address,
    source_address,
    mac_service_data_unit,
    priority,
    drop_eligible,
    frame_check_sequence,
    service_access_point_identifier,
    connection_identifier
)
```

The **destination_address** parameter is the address of an individual MAC entity or a group of MAC entities. The **source_address** parameter is the individual address of the source MAC entity. The **mac_service_data_unit** parameter is the service user data. The default **priority** value is 0. Values 1 through 7 form an ordered sequence of user_priorities, with 1 being the lowest value and 7 the highest.

The **drop_eligible** parameter provides guidance to the recipient of the service indication or of an indication corresponding to the service request, and takes the values True or False. If drop_eligible is True, the parameters of the indication should be discarded in preference to others with drop_eligible False that result in frames queued with the same traffic class. The default **drop_eligible** value is False.

The **frame_check_sequence** parameter is explicitly provided with the M_UNITDATA.indication so that it can be used in a related M_UNITDATA.request. The parameter comprises the FCS value and sufficient information to determine whether the FCS value can be used. If the frame_check_sequence parameter is provided with an M_UNITDATA.request and the receiving and the transmitting service providers

- a) Use the same algorithm to determine the FCS; and

- b) Apply that algorithm to the same fields of the frame, i.e., the FCS coverage is the same; and
- c) The data that is within the coverage of the FCS remains the same;

the transmitting service provider should use the supplied FCS value (6.5.7, 6.8).

NOTE 2—There are two possibilities for recreating a valid FCS. The first is to generate a new FCS by algorithmically modifying the received FCS, based on knowledge of the FCS algorithm and the transformations that the frame has undergone between reception and transmission. The second is to rely on the normal MAC procedures to recalculate the FCS for the outgoing frame. The former approach can be preferable in terms of its ability to protect against increased levels of undetected frame errors. Annex F of IEEE Std 802.1D-2004 discusses these possibilities in more detail. The `frame_check_sequence` parameter of the Enhanced Internal Sublayer Service (6.8) is able to signal the validity, or otherwise, of the FCS; an unspecified value in this parameter in a data request indicates to the transmitting MAC that the received FCS is no longer valid, and the FCS must therefore be recalculated.

Unlike the other parameters of the service primitives, the **service_access_point_identifier** and **connection_identifier** are not parameters of the peer to peer service. The values of the **service_access_point_identifier** and **connection_identifier** are purely local to the system within which a given service request or service indication occurs, and are not conveyed to the communicating peer system. The values are opaque to the user of the ISS, and are not manipulated by that service user, thus permitting independent operation of entities within the MAC sublayer as well as extending capabilities by the addition of protocol shims (6.1.5). The values are not conveyed in any external protocol, including management protocols; however, management protocols can convey externally visible data related to the service access point or connection. For example, there is a one-to-one association between **service_access_point_identifier** of a Bridge Port used by the MAC Relay Entity and the port number identifying that Bridge Port in management and control protocols.

The **service_access_point_identifier** parameter always identifies the service access point at which the service indication occurs or the service request is made, in the context of the receiving or transmitting system. In the common case of direct support of the ISS by a specific MAC procedure (6.7) it identifies the attached individual LAN. When a protocol entity in the interface stack either uses or supports multiple service access points, the **service_access_point_identifier** parameter associates the service primitive with the appropriate service access point.

NOTE 3—In previous versions of this standard the **service_access_point_identifier** parameter itself was implicit and was not included in the parameter list. The parameter is made explicit in this version to facilitate the description of new protocol entities that support or use multiple service access points.

The **connection_identifier** can be null, and is ignored by any specific MAC procedures except as explicitly specified in 6.7. The **connection_identifier** is used where this standard specifically provides for efficient support of a single service access point by a number of connections, i.e., by dynamically created connectivity associations between peer entities (6.1.8). For example, a Provider Instance Port (6.10) creates connections with peer Provider Instance Ports, and uses the **connection_identifier** to associate the backbone MAC address of the peer Provider Instance Port with Customer MAC Addresses that can be reached through that peer Provider Instance Port (26.4). Following an `M_UNITDATA.indication` at a given service access point with a given **source_address** and **connection_identifier**, a subsequent `M_UNITDATA.request` at the same service access point with that source address as its **destination_address** and with the same **connection_identifier** will result in an `M_UNITDATA.indication` at the peer entity selected by the **connection_identifier** (in the absence of frame loss or reconfiguration of network components). The value of a **connection_identifier** is significant only at a single service access point. Any protocol entity in the interface stack that does not specify use of the **connection_identifier** assigns the **connection_identifier** value (if any) supplied with a request from the user of the protocol entity (or with an indication from the provider of the service used by the protocol entity) to the **connection_identifier** on associated requests made (or indications generated) by the protocol entity.

NOTE 4—The ISS specification in this standard differs from that in IEEE Std 802.1D as it omits the `frame_type` and `access_priority` parameters. The `frame_type` is not required as the receipt of a frame other than a user data frame does not

cause a data indication, nor are such frames transmitted by the media independent bridge functions. The mapping of the ISS to particular access methods specified by this standard includes derivation of the access_priority parameter (for those media that require it) from the ISS priority parameter.

6.6.2 Status parameters

The Internal Sublayer Service also makes available status parameters that reflect the operational state and administrative controls over each instance of the service provided.

The **MAC_Enabled** parameter is TRUE if use of the service is permitted; and is otherwise FALSE. The value of this parameter is determined by administrative controls specific to the entity providing the service, as specified in 6.7.

The **MAC_Operational** parameter is TRUE if the entity providing the service is capable of transmitting and receiving frames and its use is permitted by management, i.e., MAC_Enabled is also TRUE. Its value is otherwise FALSE. The value of this parameter is determined by the specific MAC procedures, as specified in 6.7.

NOTE—These status parameters provide a common approach across MACs for handling the fact that:

- a) A MAC can inherently be working or not;
- b) If the MAC is working, its operational state can be administratively overridden.

6.6.3 Point-to-point parameters

The Internal Sublayer Service also makes available status parameters that reflect the point-to-point status of each instance of the service provided and provide administrative control over the use of that information.

If the **operPointToPointMAC** parameter is TRUE, the service is used as if it provides connectivity to at most one other system; if FALSE, the service is used as if it can provide connectivity to a number of systems.

The **adminPointToPointMAC** parameter can take one of three values. If it is

- a) **ForceTrue**, operPointToPointMAC shall be TRUE, regardless of any indications to the contrary generated by the service providing entity.
- b) **ForceFalse**, operPointToPointMAC shall be FALSE.
- c) **Auto**, operPointToPointMAC is determined by the service providing entity, as specified in 6.7.

The value of operPointToPointMAC is determined dynamically; i.e., it is reevaluated whenever adminPointToPointMAC or the status of the service providing entity changes.

6.6.4 Stream Reservation Protocol (SRP) Domain status parameters

A Stream Reservation Protocol (SRP) domain is a set of stations (end stations and/or Bridges), their Ports, and the attached individual LANs, that satisfy all of the following conditions for a given SR class:

- a) Those stations that transmit streams all support the credit-based shaper algorithm, defined in 8.6.8, as the transmission selection method for the SR class.
- b) The stations all support SRP, as defined in Clause 35, as the means of creating bandwidth reservations for the SR class.
- c) Those stations that transmit streams all associate the same priority value with the SR class.
- d) Each Port in the set is either an SRP domain core port or an SRP domain boundary port.
- e) Each SRP domain core Port in the set is connected, via an individual LAN that is part of the active topology, to an SRP domain core Port of another station in the set.

In Bridges that support the Stream Reservation Protocol (SRP), and for each SR class supported by the Bridge, an **SRPdomainBoundaryPort** parameter is associated with each Port of the Bridge.

Each SRPdomainBoundaryPort parameter has a Boolean value. The value of the SRPdomainBoundaryPort parameter for a given SR class is TRUE if the operation of SRP has determined that the Port is an SRP domain boundary port (3.176) for that SR class; otherwise, the Port either is an SRP domain core port (3.177) for that SR class or is not part of that SRP domain, and the SRPdomainBoundaryPort parameter value is FALSE.

6.7 Support of the Internal Sublayer Service by specific MAC procedures

This subclause specifies support of the Internal Sublayer Service by MAC Entities that use specific IEEE 802 media access methods, including the mapping to the MAC protocol and procedures for each access method, and the encoding of the parameters of the service in MAC frames. The mapping is specified by reference to the IEEE 802 standards that specify each access method. The mapping draws attention to any special responsibilities of Bridges attached to LANs of that type. MAC control frames, typically frames that control some aspect of the operation of the MAC, i.e., frames that do not convey MAC user data, do not give rise to ISS data indications and are therefore not forwarded by a Bridge to any LAN other than that on which they originated.

Each MAC Entity examines all frames received on the LAN to which it is attached. All error-free received user data frames give rise to M_UNITDATA indication primitives. A frame that is in error, as defined by the relevant MAC specification, is discarded by the MAC Entity without giving rise to any M_UNITDATA indication.

6.7.1 Support of the Internal Sublayer Service by IEEE Std 802.3 (CSMA/CD)

The CSMA/CD access method is specified in IEEE Std 802.3. Clause 3 of that standard specifies the MAC frame structure, and Clause 4 specifies the MAC method.

On receipt of an M_UNITDATA.request primitive, the local MAC Entity performs Transmit Data Encapsulation, assembling a frame using the parameters supplied as specified below. It prepends a preamble and a Start Frame Delimiter before handing the frame to the Transmit Media Access Management Component in the MAC Sublayer for transmission (4.2.3 in IEEE Std 802.3).

On receipt of a MAC frame by Receive Media Access Management, the MAC frame is passed to Receive Data Decapsulation, which validates the FCS and disassembles the frame, as specified below, into the parameters that are supplied with an M_UNITDATA.indication primitive (4.2.4 in IEEE Std 802.3).

The **destination_address** parameter is encoded in the destination address field (3.2.3 in IEEE Std 802.3).

The **source_address** parameter is encoded in the source address field (3.2.3 in IEEE Std 802.3).

The number of octets of data in the **mac_service_data_unit** parameter is either:

- a) Encoded in the Length/Type field of the MAC frame if the frame makes use of the Length interpretation of the Length/Type field (3.2.6 in IEEE Std 802.3), or
- b) Determined from the length of the received MAC frame, if the frame makes use of the Type interpretation of the Length/Type field (3.2.6 in IEEE Std 802.3).

The octets of data are encoded in the data field (3.2.7 in IEEE Std 802.3). The Length/Type field forms the initial octets of the **mac_service_data_unit** parameter.

The **priority** and **drop_eligible** parameters provided in a data request primitive are not encoded in MAC frames. The priority parameter provided in a data indication primitive shall take the value of the Default User Priority parameter for the Port through which the MAC frame was received. The default value of this parameter is 0, it may be set by management in which case the capability to set it to any of the values 0 through 7 shall be provided.

The **frame_check_sequence** parameter is encoded in the FCS field of the MAC frame (3.2.8 in IEEE Std 802.3). The FCS is computed as a function of the destination address, source address, length, data, and PAD fields. If an M_UNITDATA.request primitive is not accompanied by this parameter, it is calculated in accordance with 3.2.8 in IEEE Std 802.3.

NOTE—Since the PAD field, if present, contributes to the FCS, this parameter needs to include at least the contribution of the PAD field to the FCS in order for the original FCS to be preserved. [See Annex F in IEEE Std 802.1D-2004.]

No special action, above that specified for the support of use of the MAC Service by LLC, is required for the support of the MAC Internal Sublayer Service by the CSMA/CD access method.

The values of the MAC_Enabled and MAC_Operational parameters are determined as follows:

- c) For a MAC entity that contains a Link Aggregation sublayer, the value of MAC_Enabled is directly determined by the value of the aAggAdminState attribute (6.3.1.1.13 in IEEE Std 802.1AX-2008), and the value of MAC_Operational is directly determined by the value of the aAggOperState attribute (6.3.1.1.14 in IEEE Std 802.1AX).
- d) Otherwise, for IEEE 802.3 MAC entities that support the MAU managed Object Class (30.5.1 in IEEE Std 802.3):
 - 1) The value of MAC_Enabled is TRUE.
 - 2) The value of MAC_Operational is TRUE if the attribute aMediaAvailable carries the value *available*.
 - 3) The value of MAC_Operational is FALSE if the attribute aMediaAvailable carries any value other than *available*.
- e) Otherwise:
 - 1) The value of MAC_Enabled is TRUE.
 - 2) The value of MAC_Operational is TRUE.

From the point of view of determining the value of operPointToPointMAC (6.6.3), the MAC is considered to be connected to a point-to-point LAN if any of the following conditions are true:

- f) The MAC entity concerned contains a Link Aggregation sublayer, and the set of physical MACs associated with the Aggregator are all aggregatable; or
- g) The MAC entity concerned supports auto negotiation (Clause 28 of IEEE Std 802.3), and the auto negotiation function has determined that the LAN is to be operated in full duplex mode; or
- h) The MAC entity has been configured by management means for full duplex operation.

Otherwise, the MAC is considered to be connected to a LAN that is not point-to-point.

On receipt of an M_UNITDATA.request primitive that represents a tagged frame, the implementation is permitted to adopt either of the following approaches with regard to the operation of Transmit Data Encapsulation for frames whose length would, using the procedure as described earlier, be less than 68 octets:

- i) Use the procedure as described in a) and b). This procedure can result in tagged frames of less than 68 octets (but at least 64 octets) being transmitted; or
- j) Include additional octets before the FCS field in order for the transmitted frame length for such frames to be 68 octets. This procedure results in a minimum tagged frame length of 68 octets.

When a tagged frame of less than 68 octets in length is received on a CSMA/CD LAN, and is forwarded as an untagged frame, the procedure as described in a) and b) results in additional octets being included before the FCS field on transmission in order that the transmitted frame length meets the minimum frame size requirements of 3.2.7 in IEEE Std 802.3.

6.7.2 Support by IEEE Std 802.11 (Wireless LAN)

The wireless LAN access method is specified in IEEE Std 802.11, 1999 Edition. Clause 7 of that standard specifies frame formats, Clause 9 specifies the MAC sublayer function, and Clause 11 specifies the mandatory MAC sublayer management function.

A Bridge to an IEEE 802.11 LAN shall connect to an IEEE 802.11 Portal, which in turn connects to an IEEE 802.11 Distribution System. For the purposes of bridging, the service interface presented at the Portal is identical to the service interface presented at the IEEE 802.11 MAC SAP. An instance of an 8802-11 Distribution System can be implemented from IEEE 802 LAN components. IEEE 802.11 STAs attach to the Distribution System via an IEEE 802.11 Access Point. A bridge shall not connect to an IEEE 802.11 Independent BSS. For a description of the IEEE 802.11 architecture, see Clause 5 of IEEE Std 802.11.

On receipt of an M_UNITDATA.request primitive, the portal constructs a MAC Service Data Unit and passes it to the MAC Data service for transmission (in accordance with the frame formats and procedures specified in IEEE Std 802.11 Clauses 6, 7, 9, and Annex C) using the parameters supplied as specified below.

On receipt of a valid MAC Service Data Unit (see IEEE Std 802.11 Clauses 6, 7, 9, and Annex C), the portal generates an M_UNITDATA.indication primitive with parameter values derived from the frame fields as specified below.

When processing MSDU_from LLC, the Type subfield of the Frame Control field specified in 7.1.3.1 of IEEE Std 802.11 shall be encoded as *Data* in MAC frames (see Table 1 in IEEE Std 802.11).

The destination_address parameter is encoded in MAC frames as the DA described in Table 4 of 7.2.2 of IEEE Std 802.11.

The source_address parameter is encoded in MAC frames as the SA described in Table 4 of 7.2.2 of IEEE Std 802.11.

The mac_service_data_unit parameter is encoded in the Frame Body field (IEEE Std 802.11, 7.1.3.5) of MAC frames. The length of the MSDU shall be ≤ 2304 octets. The length is not encoded in MAC frames; rather, it is conveyed in the PHY headers.

The user_priority parameter is not encoded in MAC frames. The user_priority parameter provided in an M_UNITDATA.indication primitive shall take the value of the Default_User_Priority parameter for the port through which the MAC Service Data Unit was received. The default value of this parameter is 0, it may be set by management, in which case the capability to set it to any of the values 0 through 7 shall be provided.

The Frame Check Sequence (FCS) field of MAC frames is calculated and encoded in accordance with IEEE Std 802.11, 7.1.3.6.

No special action, above that specified in IEEE Std 802.11, is required for the support of the MAC Internal Sublayer Service by the wireless LAN access method.

6.7.3 Support by IEEE Std 802.17 (RPR)

The resilient packet ring (RPR) MAC access method is specified in IEEE Std 802.17. Clause 6 of that standard specifies the MAC service interface and reference model. Clause 7 specifies the MAC transmission and reception procedures. Clause 9 specifies the MAC frame structure.

On receipt of an M_UNITDATA.request primitive, the local MAC entity performs transmit data encapsulation, which assembles a MAC frame (IEEE Std 802.17, Clause 9) with the parameters supplied as specified in the paragraphs that follow.

On receipt of a valid MAC frame (IEEE Std 802.17, Clause 9), an M_UNITDATA.indication primitive is generated, with parameter values derived from the frame fields as specified in the paragraphs that follow.

The **destination_address** parameter is encoded in the *da* field of the MAC frame (IEEE Std 802.17, 9.2.2.3). For request primitives from a client (e.g., MAC Relay Entity) where the **source_address** parameter does not equal the MAC's address, the **destination_address** parameter is encoded in both the *da* and the *daExtended* fields of the MAC frame (IEEE Std 802.17, 9.2.2.8).

The **source_address** parameter is encoded in the *sa* field of the MAC frame (IEEE Std 802.17, 9.2.2.4) when supplied in the request primitive, and when the **source_address** is equal to the MAC's address. When the **source_address** is supplied in the request primitive, and when the **source_address** is not equal to the MAC's address, then the **source_address** is encoded in the *saExtended* field of the MAC frame (IEEE Std 802.17, 9.2.2.9).

The **mac_service_data_unit** parameter is the service user data that includes the protocol type and is encoded in the *protocolType* and *serviceDataUnit* fields of the MAC frame (IEEE Std 802.17, 9.2.2.10, 9.2.2.11).

The **priority** parameter provided in the data request primitive is encoded into the service class (*sc*) subfield of the *baseControl* field (IEEE Std 802.17, 9.6.4) of the MAC frame. This encoding is done in accordance with the priority to MAC service class mapping shown in Table 6-1.

Table 6-1—Priority to MAC service class mapping

priority	MAC service class
0	classC
1	classC
2	classC
3	classC
4	classB
5	classB
6	classA
7	classA

In the case of the indication primitive, the **priority** parameter is directly derived from the *sc* subfield of the *baseControl* field of the MAC frame. The mapping between the service class and the priority parameter of the indication primitive is provided in Table 6-2.

Table 6-2—MAC service class to priority mapping

MAC service class	priority
classC	0
classB	4
classA	6

The **frame_check_sequence** parameter found in the data request primitive is encoded in the *fcs* field of the MAC frame (IEEE Std 802.17, 9.2.2.12). The *fcs* is calculated as a 32-bit CRC starting from the first byte following the header checksum field (*hec*) (IEEE Std 802.17, 9.2.2.7) to the end of the payload (IEEE Std 802.17, 9.2.2.11) in accordance with IEEE Std 802.17, E.2. If an M_UNITDATA.request primitive is not accompanied by this parameter, it is calculated in accordance with IEEE Std 802.17, E.2.

No special action, above that specified in IEEE Std 802.17, is required for the support of the MAC Internal Sublayer Service by the RPR access method.

The 802.17 MAC service interface supports a number of optional parameters that are specific to the 802.17 MAC. These parameters take on default values in M_UNITDATA.request primitive during transmission, and they are ignored by the MAC Relay on reception. The default values and procedures for handling RPR specific parameters are defined in 6.4.1, Clause 7, and F.3.1 of IEEE Std 802.17.

6.7.4 Support by IEEE 802.20 Mobile Broadband Wireless Access Method (MBWA)

6.7.4.1 Support by Wideband mode of IEEE Std 802.20 (MBWA)

The Mobile Broadband Wireless Access Method for the 802.20 Wideband Mode is specified in Clause 5.4 and Clause 6 through Clause 17 of IEEE Std. 802.20. Clause 8 of the standard specifies the Wideband Mode Lower MAC Layer Frame structure and protocol procedures. Clause 7 specifies the Radio Link Sublayer protocol and Clause 6 defines the Services Sublayer of the Wideband Mode. Clause 11 defines the Connection Control Plane which controls the state of the air-link by managing the states of individual Lower MAC Layer protocols, and by providing individual Lower MAC Layer protocols with operating parameters.

The Basic Packet Consolidation Protocol (IEEE 802.20 subclause 8.2) provides packet consolidation on the transmit side and provides packet de-multiplexing on the receive side. It provides an interface for the Radio Link Sublayer to transport user information from the Services Sublayer.

For packets to be transmitted over the air interface (wireless medium) from either the Access Node (AN) or Access Terminal (AT), the Lower MAC Sublayer shall accept Radio Link Subayer data and control packets and shall generate Lower MAC Sublayer control packets of its own. For packets leaving the air interface (wireless medium) for the AN or AT, the Lower MAC Sublayer shall de-multiplex the received packets and shall deliver the payload to the Radio Link Sublayer. The Radio Link Sublayer shall deliver the payload to the Services Sublayer which includes support for different IEEE802.3 frame based protocols.

6.7.4.1.1 Support for Internal Sublayer Service under Wideband Mode of IEEE Std 802.20

The **destination_address**, **source_address**, **mac_service_data_unit**, and **user_priority** parameters of the M_UNITDATA primitive are encoded as described in 6.6.1.

The value of **operPointToPointMAC** (6.6.3) shall be TRUE.

The value of **MAC_Enabled** shall be determined by the procedure described in 6.6.2.

After the 802.20 AT has registered with the AN, authenticated, and performed capabilities negotiation, and after the stream is established to carry 802 frames, then the value of the **MAC_Operational** parameter shall be determined by the procedure described in 6.6.2. Beforehand, the value of **MAC_Operational** shall be FALSE.

Frame size limits are determined by IEEE Std 802.3.

6.7.4.2 Support by 625k-MC mode of IEEE Std 802.20 (MBWA)

The Mobile Broadband Wireless Access Method for 625k-MC mode is specified in Clause 5.5, Clause 18 through Clause 31, and Annex A of IEEE Std. 802.20. Clause 19 of the standard specifies 625k-MC Mode MAC Frame structure. Clause 23 specifies the MAC Protocol Sublayer function to implement the 625k-MC mode MAC service. Clause 25 specifies the L3 protocol and Clause 26 defines all the primitives used in 625k-MC Mode.

The L3 protocol layer is made up of components with distinct roles in supporting a connection across the air interface. The L3 Connection management (CM) module provides an application level interface to the higher layer. The L3 protocol creates logical connections to transport the higher layer L4 data packets. The L3 Registration Management (RM) module takes the L4 data packets provided by the higher layer (through L3 CM) and converts them into a form that can be sent over the air interface. On the receiving side, L3 RM converts packets received from the air interface back into network packets before giving them to L3 CM.

Clause 26 defines the higher layer to L3 CM Interface Primitives for the service access point that shall be provided by L3 CM for the use of the higher layer. Clause 26 defines L3 CM to L4 Interface Primitives for the service access point provided by the higher layer for the use of L3 CM.

For packets entering air interface (wireless medium) from either BS network or End User Device (EUD), L3 shall accept L4 data and L4 control packets and shall generate L3 control packets of its own, and shall then send them to L2 RLC. For packets leaving air interface (wireless medium) for BS network or EUD, L3 shall accept byte streams from L2 RLC, shall determine whether the packet is a data packet, an L3 control packet, or an L4 control packet, and shall route the L4 control and data packets to the higher layer.

6.7.4.2.1 Support for Internal Sublayer Service under 625k-MC Mode of IEEE Std 802.20

The **destination_address**, **source_address**, **mac_service_data_unit**, and **user_priority** parameters of the M_UNITDATA primitive are encoded as described in 6.6.1. and presented as an ISS supported IEEE802.3 MAC to the higher layer. The higher layer triggers the L3 protocol of 625k-MC. The L3 CM module state machine shall respond to requests from the higher layer for virtual connections across the air interface, and requests registrations from the L3 RM to allow the virtual connections to use physical channels (streams).

The value of **operPointToPointMAC** (6.6.3) shall be TRUE.

The value of **MAC_Enabled** shall be determined by the procedure described in 6.6.2.

Initially, the value of **MAC_Operational** shall be FALSE. After the UT has registered with the BS, authenticated, and performed capabilities negotiation, and after the stream is established to carry 802 frames, then the value of the **MAC_Operational** parameter shall be determined by the procedure described in 6.6.2. Frame size limits are determined by IEEE Std 802.3.

6.8 Enhanced Internal Sublayer Service

The EISS is derived from the ISS (see 6.6) by augmenting that specification with elements necessary to the operation of the tagging and untagging functions of a VLAN-aware Bridge (3.200). Within the attached end station, these elements can be considered either to be below the MAC Service boundary, and pertinent only to the operation of the service provider, or to be local matters not forming part of the peer-to-peer nature of the MAC Service.

The EISS provides the same service status and point-to-point parameters as the ISS (6.6.2, 6.6.3).

6.8.1 Service primitives

The unit-data primitives that define this service are

```
EM_UNITDATA.indication      (
    destination_address,
    source_address,
    mac_service_data_unit,
    priority,
    drop_eligible,
    vlan_identifier,
    frame_check_sequence,
    service_access_point_identifier,
    connection_identifier,
)
EM_UNITDATA.request          (
    destination_address,
    source_address,
    mac_service_data_unit,
    priority,
    drop_eligible,
    vlan_identifier,
    frame_check_sequence,
    service_access_point_identifier,
    connection_identifier,
)
```

The **destination_address**, **source_address**, **mac_service_data_unit**, **priority**, **drop_eligible**, **service_access_point_identifier**, **connection_identifier**, and **frame_check_sequence** parameters are as defined for the ISS.

NOTE 1—Some of the functions supporting the EISS may result in changes to the **mac_service_data_unit** or other parameters used to construct a frame. The original FCS associated with a frame is invalidated if there are changes to any fields of the frame, if fields are added or removed, or if bit ordering or other aspects of the frame encoding have changed. An invalid FCS is signaled in the EISS by an unspecified value in the **frame_check_sequence** parameter. This signals the need for the FCS to be regenerated according to the normal procedures for the transmitting MAC. The options for regenerating the FCS under these circumstances are discussed in Annex F of IEEE Std 802.1D-2004.

The **vlan_identifier** parameter carries the VLAN identifier (VID).

6.8.2 Status parameters

The EISS also makes available the **MAC_Enabled** and **MAC_Operational** status parameters that reflect the operational state and administrative controls over each instance of the service provided. The values of these parameters are mapped directly from the values made available by the ISS (6.6.2).

6.8.3 Point-to-point parameters

The EISS also makes available the **operPointToPointMAC** and **adminPointToPointMAC** status parameters that reflect the point-to-point status of each instance of the service provided and provide administrative control over the use of that information. The values of these parameters are mapped directly from the values made available by the ISS (6.6.3).

6.9 Support of the EISS

The EISS is supported by tagging and detagging functions that in turn use the ISS (6.6, 6.7). Any given instance of the EISS shall be supported by using one but not both of the following VLAN tag types:

- a) Customer VLAN tag (C-TAG); or
- b) Service VLAN tag (S-TAG);

selected as specified in 9.5.

Each Bridge Port shall support the following parameters for use by these functions:

- c) an Acceptable Frame Types parameter with at least one of the following values:
 - 1) *Admit Only VLAN-tagged frames*;
 - 2) *Admit Only Untagged and Priority-tagged frames*;
 - 3) *Admit All frames*.
- d) a PVID parameter for port-based VLAN classification;

and may support the following parameter:

- e) a VID Set for Port-and-Protocol-based classification (6.12).

An instance of the EISS supported by using the S-VLAN tag type may also support the following parameter:

- f) A VID translation table.

All three values for Acceptable Frame Types may be supported; if so, they shall be configurable using the management operations defined in Clause 12, and the default shall be *Admit All Frames*. A frame is treated as Untagged if the initial octets of the **mac_service_data_unit** parameter do not contain a VLAN tag of the type used to support the EISS (9.5), as Priority-tagged if the value contained in the VID field of the VLAN tag is zero, and as VLAN-tagged if the value is nonzero.

The PVID and VID Set shall contain valid VID values (Table 9-2) and may be configured by management. If they have not been explicitly configured, the PVID shall assume the value of the default PVID defined in Table 9-2 and the VID Set shall be empty.

NOTE—The default behavior of a Bridge that supports Port-and-Protocol-based classification is the same as that of a Bridge that supports only port-based classification, since all the Protocol Group Identifiers in the VID Set for each Port assign the same VID as the PVID.

The VID translation table, when supported, shall contain a one-to-one bidirectional mapping of VID values included in the S-TAG of frames transmitted and received on the port (local VID) and VID values passed in the parameters of the EISS service primitives (relay VID). The table is configurable by management, and the default table configuration maps each value of the local VID to the same value for the relay VID.

6.9.1 Data indications

On receipt of an M_UNITDATA.indication primitive from the Internal Sublayer Service, the received frame is discarded if:

- a) The initial octets of the mac_service_data_unit do not contain a valid VLAN tag header (9.3) of the type used to support the EISS (9.5), and the Acceptable Frame Types is *Admit Only VLAN-tagged frames*; or,
- b) The initial octets of the mac_service_data_unit contain a valid VLAN tag header (9.3) of the type used to support the EISS (9.5), and
 - 1) The VID value is FFF, reserved in Table 9-2 for future implementation use; or
 - 2) The VID value is 000 (indicating priority-tagged), and the Acceptable Frame Types is *Admit Only VLAN-tagged frames*; or
 - 3) The VID value is in the range 001-FFE, and the Acceptable Frame Types is *Admit Only Untagged and Priority-tagged frames*.

Otherwise an EM_UNITDATA.indication primitive is invoked, with parameter values determined as follows:

The **destination_address**, **source_address**, and **frame_check_sequence** parameters carry values equal to the corresponding parameters in the received data indication. The values of the remaining parameters are affected by the contents of the tag header if the frame contains a VLAN tag of the type supported by this instance of the EISS.

The value of the **mac_service_data_unit** parameter is as follows:

- c) If the frame is tagged, then the value used is equal to the value of the received mac_service_data_unit following removal of the tag header.
- d) Otherwise, the value used is equal to the value of the received mac_service_data_unit.

The value of the **vlan_identifier** parameter is as follows:

- e) The value contained in the VID field, optionally translated using the VID translation table, if the frame is VLAN-tagged;
- f) The value of the PVID for the Port, if the frame is untagged or Priority-tagged and port-and-protocol VLAN classification is not implemented;
- g) As determined by Port-and-Protocol-based VLAN classification (6.12) if that capability is implemented and the frame is untagged or Priority-tagged.

The value of the **drop_eligible** and **priority** parameters are determined as follows:

- h) If the frame is tagged, the value of the drop_eligible parameter and the received priority value are decoded from the tag header as described in 6.9.3. Otherwise;
- i) The received priority value and the drop_eligible parameter value are the values in the M_UNITDATA.indication.
- j) The value of the priority parameter is then regenerated from the received priority, as specified in 6.9.4.

6.9.2 Data requests

On invocation of an EM_UNITDATA.request primitive by a user of the EISS, an M-UNITDATA.request primitive is invoked, with parameter values as follows:

The **destination_address**, **source_address**, **drop_eligible**, and **priority** parameters carry values equal to the corresponding parameters in the received data request.

Unless the Bridge Port is a member of the untagged set (8.8.2) for the VID identified by the **vlan_identifier** parameter, then a tag header, formatted as necessary for the destination MAC type, is inserted as the initial octets of the **mac_service_data_unit** parameter. The values of the **vlan_identifier** (optionally translated using the VID translation table), **priority**, and **drop_eligible** parameters are used to determine the contents of the tag header, in accordance with Clause 9.

If the Bridge Port is a member of the untagged set (8.8.2) for the VID identified by the **vlan_identifier** parameter, no tag header is inserted.

The remaining octets of the **mac_service_data_unit** parameter are those accompanying the EISS-request. If the data request is a consequence of relaying a frame and the MAC type of the Port differs from that used to receive the frame, they are modified, if necessary, in accordance with the procedures described in ISO/IEC 11802-5, IETF RFC 1042 (1988), and IETF RFC 1390.

The value of the **frame_check_sequence** parameter is determined as follows:

- a) If the **frame_check_sequence** parameter received in the data request either is unspecified or still carries a valid value, then that value is used.
- b) Otherwise, the value used is either unspecified or derived from the received FCS information by modification to take account of the conditions that have caused it to become invalid.

6.9.3 Priority Code Point encoding

The priority and drop_eligible parameters are encoded in the Priority Code Point (PCP) field of the VLAN tag using the Priority Code Point Encoding Table for the Port, and they are decoded from the PCP using the Priority Code Point Decoding Table. For each Port, the Priority Code Point Encoding Table has 16 entries, corresponding to each of the possible combinations of the eight possible values of priority (0 through 7) with the two possible values of drop_eligible (True or False). For each Port, the Priority Code Point Decoding Table has 8 entries, corresponding to each of the possible PCP values.

Alternative values for each table are specified as rows in Table 6-3 and Table 6-4, with each alternative labeled by the number of distinct priorities that can be communicated, and the number of these for which drop precedence can be communicated. For example, the table entries 6P2D allow six distinct priorities with drop precedence for two. The default values (8P0D) specify a PCP value equal to the priority value and do not support communication of drop precedence in the PCP field. The combination of the priority and drop_eligible parameters is shown by the priority alone if drop_eligible is False, and by the priority followed by the letters “DE” if drop_eligible is True.

If the VLAN tag is a Service VLAN tag, then Table 6-3 and Table 6-4 shall be supported. If the VLAN tag is a Customer VLAN tag, then the 8P0D row of each table shall be supported, and the remaining rows may be supported.

NOTE—To avoid potential misordering of packets, all Ports on a LAN should select the same row of Table 6-3 and Table 6-4. Interoperability considerations with Bridges supporting only the 8P0D row are discussed in I.8.

Table 6-3—Priority Code Point encoding

priority drop_eligible	7	7DE	6	6DE	5	5DE	4	4DE	3	3DE	2	2DE	1	1DE	0	0DE	
PCP	8P0D (default)	7	7	6	6	5	5	4	4	3	3	2	2	1	1	0	0
7P1D	7	7	6	6	5	4	5	4	3	3	2	2	1	1	0	0	
6P2D	7	7	6	6	5	4	5	4	3	2	3	2	1	1	0	0	
5P3D	7	7	6	6	5	4	5	4	3	2	3	2	1	0	1	0	

Table 6-4—Priority Code Point decoding

PCP	7	6	5	4	3	2	1	0	
priority drop_eligible	8P0D (default)	7	6	5	4	3	2	1	0
7P1D	7	6	4	4DE	3	2	1	0	
6P2D	7	6	4	4DE	2	2DE	1	0	
5P3D	7	6	4	4DE	2	2DE	0	0DE	

The values in the Priority Code Point Encoding Table and Priority Code Point Decoding Table may be modified by management, as described in Clause 12. If this capability is provided, the value of the table entries shall be independently settable for each Port. The values shall be constrained as follows:

- a) For any two priority values with the same drop_eligible value, the lower priority shall not map to a higher PCP than the higher priority.
- b) The lower of any two PCP values shall not map to a higher priority than the higher PCP.
- c) With the exception of values 0 and 1, any priority value combined with drop_eligible True shall not map to a higher PCP than the same priority value combined with drop_eligible False.
- d) With the exception of values 0 and 1, any PCP value shall not map to a priority value combined with drop_eligible True if a lower PCP maps to the same priority value combined with drop_eligible False.

The values may be further constrained, but if the tables can be modified, the default values shown in Table 6-3 and Table 6-4, and the alternative sets of values in Table 6-3 and Table 6-4 shall be settable.

The drop_eligible parameter may also be encoded in and decoded from the Drop Eligible Indicator (DEI) in the VLAN tag. Use of the DEI allows the VLAN tag to convey eight distinct priorities, each with a drop eligible indicator. If this capability is provided, it shall be independently manageable for each Port by means of a Use_DEI parameter. If the Use_DEI is True for the Port, the drop_eligible parameter is encoded in the DEI of transmitted frames, and the drop_eligible parameter shall be True for a received frame if the DEI is set in the VLAN tag or the Priority Code Point Decoding Table indicates drop_eligible True for the received PCP value. If the Use_DEI parameter is False, the DEI shall be transmitted as zero and ignored on receipt. The default value of the Use_DEI parameter is False.

6.9.4 Regenerating priority

The priority of each received frame is regenerated using priority information contained in the frame and the Priority Regeneration Table for the reception Port. For each reception Port, the Priority Regeneration Table has eight entries, corresponding to the eight possible values of priority (0 through 7). Each entry specifies, for the given value of received priority, the corresponding regenerated value.

For untagged frames, the priority parameter signaled in the corresponding M_UNITDATA.indication is taken to be the received priority. For tagged frames, the priority signaled in the PCP field of the tag header is taken to be the received priority.

NOTE 1—IEEE 802 LAN technologies signal a maximum of eight priority values. I further explains the use of priority values and how they map to traffic classes.

Table 6-5 specifies default regenerated priority values for each of the eight possible values of the received priority. These default values shall be used as the initial values of the corresponding entries of the Priority Regeneration Table for each Port.

The values in the Priority Regeneration Table may be modified by management, as described in Clause 12. If this capability is provided, the value of the table entries shall be independently settable for each reception Port and for each value of received priority, and the Bridge shall have the capability to use the full range of values in the parameter ranges specified in Table 6-5.

NOTE 2—The regeneration and mapping of priority within the Bridge should be consistent with the end-to-end significance of that priority across the rest of the Bridged Local Area Network. The regenerated priority value is used:

- Via the traffic class table (8.6.6) to determine the traffic class for a given outbound Port, and
- Via fixed, MAC type-specific mappings (6.7) to determine the access priority that will be used for certain media access methods.

Table 6-5—Priority regeneration

Received priority	Default regenerated priority	Range
0	0	0–7
1	1	0–7
2	2	0–7
3	3	0–7
4	4	0–7
5	5	0–7
6	6	0–7
7	7	0–7

In Bridges that support SRP, an SRP domain boundary port priority regeneration override table is maintained for each reception Port. This table associates a priority value with each SR class supported by the Bridge, and specifies the priority value to be used as the regenerated priority, in preference to the value in the

Priority Regeneration table, when the SRPdomainBoundaryPort parameter for that SR class has the value TRUE. Table 6-6 specifies the default values for priority and regenerated priority for two SR classes, SR class A and SR class B. In cases where only SR class B is supported, only the default value shown in the table for SR Class B is used.

Table 6-6—Default SRP domain boundary port priority regeneration override values

SR class	Default priority	Default regenerated priority for SRP domain boundary Ports	Range
A	3	0	0–7
B	2	0	0–7

The priority values contained in the SRP domain boundary port priority regeneration override table may be modified by management, as described in Clause 12. If this capability is provided, the value of the table entries shall be independently settable for each reception Port and for each supported SR class, and the Bridge shall have the capability to use the full range of values in the parameter ranges specified in Table 6-6.

NOTE 3—The default values specified for the SRP domain boundary port Priority Regeneration Table map inbound PCP values onto the next lowest priority that is available, leaving the remaining priority values unchanged. The consequence of this remapping is that traffic entering an SRP domain that carries the priority value associated with that domain is forwarded by the bridge at the domain boundary using a lower priority. There is no means of mapping these priority values back to their original values as frames exit from an SRP domain. The default mappings are based on the assumption that normally, two SR classes (A and B) are used to support traffic for which bandwidth has been reserved; if only the lower class (B) is supported, then the default override value for the unsupported A class does not apply.

NOTE 4—The ability for the priority regeneration table to be freely modified by management means that it is possible to create mappings that result in reserved and nonreserved traffic sharing the same priority. This will have unpredictable effects upon the behavior of the network with respect to the worst-case latency that can be assumed for reserved traffic.

6.10 Support of the ISS/EISS by Provider Instance Ports

Each Provider Instance Port (PIP) on a Backbone Edge Bridge uses a single ISS-SAP (6.1.4, 6.6) and the Backbone Service Instance tag (9.5) to provide a separate Bridge Port (the Virtual Instance Port) for each backbone service instance configured for the PIP. Each Virtual Instance Port (VIP) supports either the VLAN-unaware MAC Relay Entity of a T-component or the VLAN-aware MAC Relay Entity of an I-component. A single VIP ISS-SAP is provided to a T-component. An I-component uses an EISS-SAP, supported by its own instance of the functions specified in 6.9 with the Service VLAN tag (9.5), for each VIP ISS-SAP. See Figure 6-4.

The EISS of each Virtual Instance Port (VIP-EISS) shall be supported by an instance of the functions of 6.9 using the Service VLAN tag type (9.5). The ISS of all of the Virtual Instance Ports (VIP-ISS) shall be supported by a single instance of the functions specified in this subclause using the Backbone Service Instance tag type (9.5).

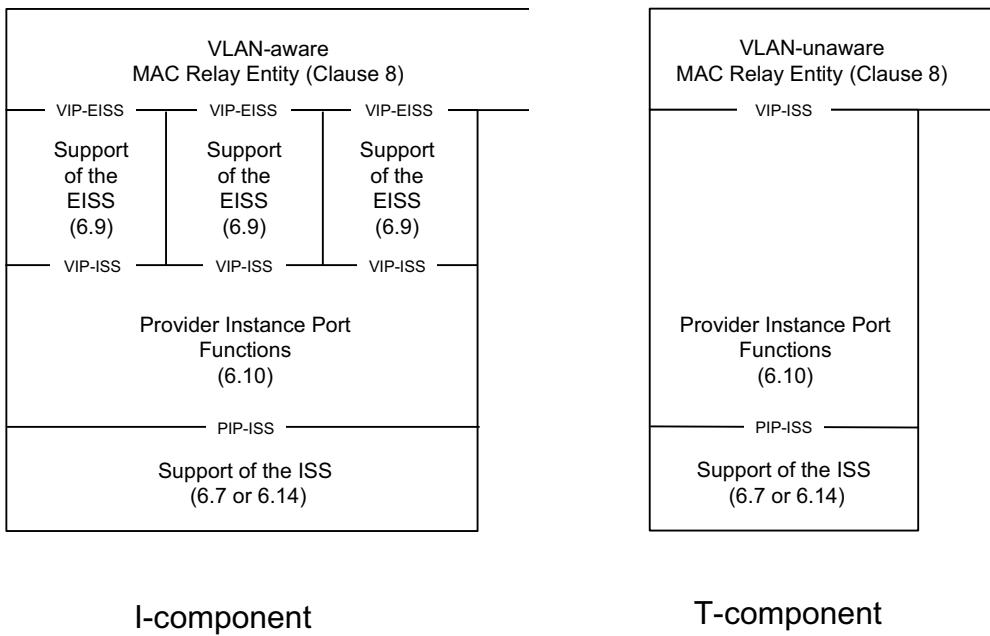


Figure 6-4—Provider Instance Ports

NOTE 1—Higher Layer Entities may interface to any Virtual Instance Port and/or the Provider Instance Port at the service access points represented by any VIP-ISS and/or the PIP-ISS using the Bridge Port Connectivity specified in 8.5.1. Protocol shims such as Connectivity Fault Management Maintenance Points may interface at any VIP-EISS using the EISS Multiplex Entity specified in 6.17, or at the PIP-ISS using the Backbone Service Instance Multiplex Entity specified in 6.18. Figure 6-4 shows a Provider Instance Port with both Bridge Port Connectivity and CFM shims.

Each Provider Instance Port shall have an individual Backbone MAC address referred to as the PIP MAC address (26.4) for use by the functions specified in this subclause.

Each Virtual Instance Port shall support the following parameters for use by these functions:

- A VIP-ISID parameter for the Backbone Service Instance Identifier of the backbone service instance associated with this Virtual Instance Port;
- A Default Backbone Destination;
- An adminPointToPointMAC;
- An enableConnectionIdentifier.

The VIP-ISID shall contain a valid I-SID value (9.7), which is configured by management.

The Default Backbone Destination parameter contains a MAC address to be used in the destination_address parameter of an M_UNITDATA.request at the PIP-ISS when a backbone destination address cannot be derived from the connection_identifier parameter of the M_UNITDATA.request from the VIP-ISS. The default value of the Default Backbone Destination is the Backbone Service Instance Group address for the VIP-ISID (generated by concatenating the Provider Backbone Bridge OUI with the VIP-ISID value as specified in 26.4).

The adminPointToPointMAC parameter of the VIP-ISS reflects the point-to-point status of the backbone service instance associated with the VIP. The default value of the adminPointToPointMAC parameter is ForceFalse. The value may be configured by management to ForceTrue when instantiating a VIP for a

point-to-point backbone service instance. A value of ForceFalse or Auto results in a `operPointToPointMAC` value of FALSE; a value of ForceTrue results in a `operPointToPointMAC` value of TRUE. Whenever the `operPointToPointMAC` parameter transitions to FALSE, the value for the Default Backbone Destination parameter is set to the Backbone Service Instance Group address for the VIP-ISID. When the `operPointToPointMAC` parameter is TRUE, the Default Backbone Destination parameter is modified by subsequent `M_UNITDATA.indications` as specified in 6.10.1 (and described in 26.4.1).

The `enableConnectionIdentifier` parameter allows the VIP to use the `connection_identifier` parameter to learn associations between a backbone MAC address and a customer MAC address. The default value is TRUE. This parameter should be configured to FALSE at the root node of a point-to-multipoint TESI.

6.10.1 Data indications

On receipt of an `M_UNITDATA.indication` primitive from the PIP-ISS, if the PIP is congestion aware (5.4.1.4) and the initial octets of the `mac_service_data_unit` contain a valid Congestion Notification Message encapsulation, the received frame is processed according to 32.16. Otherwise, the received frame shall be discarded if

- a) The initial octets of the `mac_service_data_unit` do not contain a valid Backbone Service Instance tag header (I-TAG); or
- b) The Use Customer addresses (UCA) bit in the I-TAG is not zero; or
- c) The Res2 field in the I-TAG is not zero; or
- d) The I-SID value in the I-TAG does not match the VIP-ISID parameter for one of the VIPs supported by this PIP; or
- e) The `destination_address` parameter of the received `M_UNITDATA.indication` primitive does not match one of the following:
 - 1) The PIP MAC address; or
 - 2) A Backbone Service Instance Group address; or
 - 3) The broadcast address.

NOTE 2—The generation of I-tagged frames specified in 6.10.2 and the allowed manipulation of backbone addresses specified in 6.11 never result in an I-tagged frame with a broadcast destination address. The broadcast address is included in the previous list on the principle that the broadcast address is a valid address at any reception port by definition.

The received frame may be discarded if

- f) The `source_address` parameter of the received `M_UNITDATA.indication` primitive is the PIP MAC address.

Otherwise an `M_UNITDATA.indication` primitive is invoked at the VIP-ISS with a VIP-ISID value matching the I-SID in the I-TAG. The parameter values of the indication are determined as follows:

The `destination_address` parameter contains the value of the encapsulated C-DA field of the I-TAG.

The `source_address` parameter contains the value of the encapsulated C-SA field of the I-TAG.

If `enableConnectionIdentifier` is TRUE, then the `connection_identifier` parameter references the value of the `source_address` parameter (B-SA) of the `M_UNITDATA.indication` primitive received from the PIP-ISS; otherwise the `connection_identifier` parameter is null. The `connection_identifier` parameter in the `M_UNITDATA.indication` primitive received from the PIP-ISS is ignored.

NOTE 3—The `connection_identifier` has only local significance meaning that it is used only at a specific PIP. The only requirement imposed by this subclause is that the `connection_identifier` uniquely identifies an individual backbone MAC

address. Whether the connection_identifier takes the value of that address, or a value from which that address can be determined, is an implementation decision that is not externally observable.

If the value of the operPointToPointMAC parameter of the VIP-ISS is TRUE, then the Default Backbone Destination parameter is set to the value of the source_address parameter of the M_UNITDATA.indication primitive.

The **service_access_point_identifier** parameter contains a value that uniquely identifies the VIP within the I-component. This is a local value that is used only within the I-component, and is used by the relay functions (Clause 8) as the port number of the VIP. The service_access_point_identifier parameter received in the M_UNITDATA.indication primitive is ignored.

The value of the **drop_eligible** and **priority** parameters are decoded from the I-PCP and I-DEI fields of the I-TAG as described in 6.10.3.

The value of the **mac_service_data_unit** parameter is the value of the received mac_service_data_unit following the removal of the I-TAG.

The value of the **frame_check_sequence** parameter is either unspecified or derived from the received FCS information modified to take into account the changes to the frame.

6.10.2 Data requests

On invocation of an M_UNITDATA.request primitive at any VIP-ISS, an M_UNITDATA.request primitive is invoked at the PIP-ISS, with parameter values as follows:

If enableConnectionIdentifier is TRUE, the connection_identifier is not null, and the connection_identifier references an address retained by the Provider Instance Port, then the value for the **destination_address** is the address referenced by the connection_identifier. Otherwise, the value for the **destination_address** is the contents of the Default Backbone Destination parameter of the Virtual Instance Port.

The value for the **source_address** is the PIP MAC address.

The **priority** and **drop_eligible** parameters carry the same value as the corresponding parameter in the VIP-ISS request.

The **connection_identifier** parameter is NULL.

The **mac_service_data_unit** parameter is constructed from the **mac_service_data_unit** of the M_UNITDATA.request primitive from the VIP-ISS by prepending an I-TAG as follows:

- a) The encapsulated C-SA field is the value of the source_address parameter of the VIP-ISS request;
- b) The encapsulated C-DA field is the value of the destination_address parameter of the VIP-ISS request;
- c) The UCA, Res1, and Res2 fields are all zero;
- d) The I-SID field is the value contained in the VIP-ISID parameter;
- e) The I-PCP and I-DEI fields are encoded from the priority and drop_eligible parameters of the VIP-ISS request as specified in 6.10.3.

The value of the **frame_check_sequence** parameter is either unspecified or derived from the received FCS information modified to take into account the changes to the frame.

6.10.3 Priority Code Point encoding

The encoding (and decoding) of the I-PCP and I-DEI fields from (to) the **priority** and **drop_eligible** parameters is accomplished using a Priority Code Point Encoding Table (and Priority Code Point Decoding Table) as described in 6.9.3.

6.11 Support of the EISS by Customer Backbone Ports

The functions specified in this subclause replace the functions specified in 6.9 when the EISS is supported by a Customer Backbone Port in a Backbone Edge Bridge (Clause 25). The functions specified in this subclause use a single instance of the ISS (6.6, 6.7) and provide a single instance of the EISS. The EISS shall be supported by functions using both the Service VLAN tag type (S-TAG) and the Backbone Service I-TAG as specified in 9.5.

NOTE 1—Although the Customer Backbone Port does not transmit or receive S-tagged frames, it supports the Service VLAN tag type in the sense that it is a port on an S-VLAN component and the *vlan_identifier* parameter contains a Service VLAN Identifier.

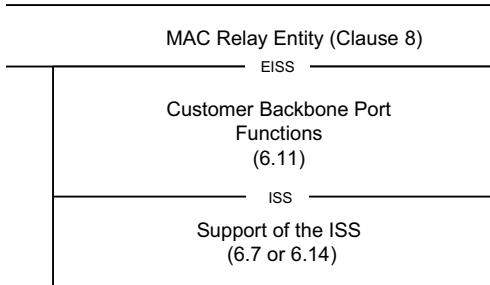


Figure 6-5—B-Component Customer Backbone Port

NOTE 2—Higher Layer Entities may interface to the Customer Backbone Port at the service access point represented by the ISS using the Bridge Port Connectivity specified in 8.5.1. Protocol shims such as Connectivity Fault Management Maintenance Points may interface at the EISS using the EISS Multiplex Entity specified in 6.17, or at the ISS using the Backbone Service Instance Multiplex Entity specified in 6.18. Figure 26-2 shows a Customer Backbone Port with both Bridge Port Connectivity and CFM shims.

The Customer Backbone Port shall support the following parameters for use by these functions:

- a) A PVID parameter for port-based B-VLAN classification;
- b) A Backbone Service Instance table.

The PVID shall contain a valid VID value (Table 9-2) and may be configured by management. If it has not been explicitly configured, the PVID shall assume the value of the default PVID defined in Table 9-2.

The Backbone Service Instance table is configurable by management and has one entry for each backbone service instance supported on the Customer Backbone Port. The Backbone Service Instance table is used to filter frames containing an I-SID value that is not configured on this Customer Backbone Port. It may also be used to provide backbone service instance specific assignment of a B-VID, to translate I-SID values, and/or to translate backbone destination addresses. Entries in the Backbone Service Instance table may be set automatically by the protection switching procedures described in 26.10.3.4. The Backbone Service Instance table shall contain the following field for each entry:

- c) A Backbone Service Instance Identifier (Backbone-SID). This contains the identifier used in the backbone network for the backbone service instance.

The Backbone Service Instance table may also contain one or more of the following additional fields for each entry:

- d) A Backbone VLAN Identifier (B-VID). If the B-VID field is supported, it shall contain a valid VID value (Table 9-2) for that backbone service instance. The default value is the same as the PVID.
- e) A Local Service Instance Identifier (Local-SID). This field is used to perform a bidirectional 1:1 mapping between the Backbone-SID (contained in the I-TAG in the **mac_service_data_unit** at the EISS) and the Local-SID (contained in the I-TAG in the **mac_service_data_unit** at the ISS). The default value of the Local-SID field is the value of the Backbone-SID field.
- f) A Default Backbone Destination. If this field is present, it is used as the **destination_address** in the EISS indication when the **destination_address** parameter of the ISS indication is the Backbone Service Instance Group address. It allows the backbone provider to replace the Backbone Service Instance Group address with an individual or group address to limit the distribution of the frame within a Backbone VLAN. The default value is the Backbone Service Instance Group address.

NOTE 3—If a CBP is associated with a TESI the Default Backbone Destination field and the Backbone VLAN identifier field is supported.

6.11.1 Data indications

On receipt of an **M_UNITDATA.indication** primitive from the Internal Sublayer Service, the received frame shall be discarded if

- a) The initial octets of the **mac_service_data_unit** do not contain a valid I-TAG; or
- b) The Res2 field in the I-TAG is not zero; or
- c) The Local-SID field of the Backbone Service Instance table is supported and the I-SID value in the I-TAG does not match the Local-SID value in any entry in the Backbone Service Instance table; or
- d) The Local-SID field of the Backbone Service Instance table is not supported and the I-SID value in the I-TAG does not match the Backbone-SID value in any entry in the Backbone Service Instance table.

Otherwise, an **EM_UNITDATA.indication** primitive is invoked at the EISS with parameter values determined using one entry in the Backbone Service Instance table. If the Local-SID field of the Backbone Service Instance table is supported then the entry with a Local-SID value that matches the I-SID value in the I-TAG is selected. Otherwise, the entry with a Backbone-SID value that matches the I-SID value in the I-TAG is selected. The parameter values are then determined as follows:

The **mac_service_data_unit** parameter is determined as follows:

- e) If the Local-SID field of the Backbone Service Instance table is supported, then value of the **mac_service_data_unit** parameter is the value of the **mac_service_data_unit** parameter of the ISS indication modified by replacing the value of the I-SID field in the I-TAG with the value of Backbone-SID field in the selected Backbone Service Instance table entry;
- f) Otherwise, the **mac_service_data_unit** parameter contains the same value as the **mac_service_data_unit** parameter of the ISS indication.

The **destination_address** parameter is determined as follows:

- g) If the **destination_address** in the ISS indication is the Backbone Service Instance Group address and the Default Backbone Destination field of the Backbone Service Instance table is supported, then the

destination_address parameter is the Default Backbone Destination value in the selected Backbone Service Instance table entry;

- h) Otherwise, if the **destination_address** in the ISS indication is the Backbone Service Instance Group address and the Local-SID field of the Backbone Service Instance table is supported, then the **destination_address** parameter is the Backbone Service Instance Group address constructed using the Backbone-SID value in the selected Backbone Service Instance table entry;
- i) Otherwise, the **destination_address** parameter contains the same value as the **destination_address** parameter of the ISS indication.

The **source_address** parameter contains the same value as the **source_address** parameter of the ISS indication.

The value of the **vlan_identifier** parameter is determined as follows:

- j) If the B-VID field of the Backbone Service Instance table is supported, then the **vlan_identifier** parameter contains the B-VID value in the selected Backbone Service Instance table entry;
- k) Otherwise, the **vlan_identifier** parameter contains the value of the PVID parameter.

The value of the **drop_eligible** and **priority** parameters are determined as follows:

- l) The value of the **drop_eligible** parameter and the **received_priority** value are decoded from the I-DEI and the I-PCP fields of the I-TAG in the **mac_service_data_unit** as described in 6.11.3;
- m) The value of the **priority** parameter is then regenerated from the **received_priority** value, as specified for the **received_priority** in 6.11.4.

The value of the **frame_check_sequence** parameter is either unspecified or derived from the received FCS information modified as necessary to take into account changes to the frame.

6.11.2 Data requests

On invocation of an EM_UNITDATA.request primitive by a user of the EISS, the frame shall be discarded if

- a) The initial octets of the **mac_service_data_unit** do not contain a valid I-TAG;
- b) The Res2 field in the I-TAG is not zero;
- c) The I-SID value in the I-TAG does not match the Backbone-SID value in any of the entries in the Backbone Service Instance table.

Otherwise, an M_UNITDATA.request primitive is invoked, with parameter values determined using the Backbone Service Instance table entry with a Backbone-SID value that matches the I-SID value in the I-TAG. The parameter values are determined as follows:

The value of the **mac_service_data_unit** parameter is determined as follows:

- d) If the Local-SID field of the Backbone Service Instance table is supported, then the **mac_service_data_unit** parameter contains the value of the **mac_service_data_unit** parameter of the EISS request modified by replacing the I-SID in the I-TAG with the Local-SID value in the selected Backbone Service Instance table entry;
- e) Otherwise, the **mac_service_data_unit** parameter contains the same value as the **mac_service_data_unit** parameter of the EISS request.

The **destination_address** parameter is determined as follows:

- f) If the Local-SID field of the Backbone Service Instance table is supported and the **destination_address** in the EISS request is the address of this Customer Backbone Port, or is a group

address, then the **destination_address** parameter is the Backbone Service Instance Group address constructed from the Local-SID value in the selected Backbone Service Instance table entry;

- g) If the Local-SID field of the Backbone Service Instance table is not supported and the **destination_address** in the EISS request is the address of this Customer Backbone Port, or is a group address, then the **destination_address** parameter is the Backbone Service Instance Group address constructed from the Backbone-SID value in the selected Backbone Service Instance table entry;
- h) Otherwise, the **destination_address** parameter contains the same value as the **destination_address** parameter of the EISS request.

The **source_address** parameter contains the same value as the **source_address** parameter of the EISS request.

The **priority** and **drop_eligible** parameters carry the same value as the corresponding parameter in the EISS request.

The value of the **frame_check_sequence** parameter is either unspecified or derived from the received FCS information modified to take into account any changes to the frame.

6.11.3 Priority Code Point decoding

The decoding of the I-PCP and I-DEI fields to the **priority** and **drop_eligible** parameters is accomplished using a Priority Code Point Decoding Table as described in 6.9.3.

6.11.4 Regenerating priority

The priority of received frames is regenerated using priority information contained in the I-PCP and the Priority Regeneration table for the reception Port. For each reception Port, the Priority Regeneration table has eight entries, corresponding to the eight possible values of priority (0 through 7). Each entry specifies, for the given value of **received_priority**, the corresponding regenerated value. The priority decoded from the I-PCP field of the I-TAG is taken to be the **received_priority**.

NOTE—IEEE 802 LAN technologies signal a maximum of eight priority values. Annex G further explains the use of priority values and how they map to traffic classes.

Table 6-5 specifies default regenerated priority values for each of the eight possible values of the **received_priority**. These default values shall be used as the initial values of the corresponding entries of the Priority Regeneration table for each Port. The table may be configured as described in 6.9.4.

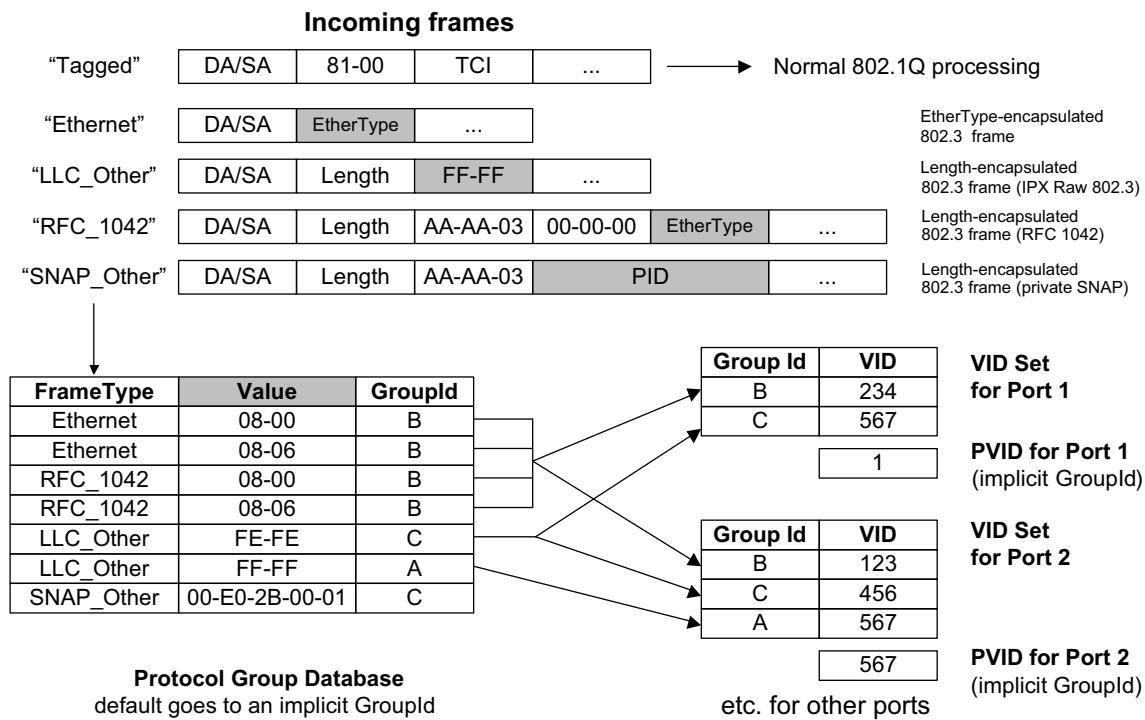
6.12 Protocol VLAN classification

Each instance of the tagging and detagging functions that supports the EISS (6.9), and implements the optional Port-and-Protocol-based VLAN classification, shall implement a VID Set, each member of which associates values of a Protocol Group Identifier (6.12.2) with a VID. Each Untagged and Priority-tagged frame received is assigned a **vlan_identifier** equal to the VID Set value for the reception Port and the Protocol Group Identifier selected by matching the received frame with a Protocol Template.

The **detagged_frame_type** parameter indicates the frame format. Its value is determined as follows:

- a) If the frame is Untagged or Priority Tagged, this parameter is present and indicates the link-layer encapsulation format of the *Detagged Frame*. The Detagged Frame of an Untagged Frame is the frame itself. The Detagged Frame of a Tagged Frame or Priority Tagged Frame is the frame that results from untagging the frame in accordance with the frame format described in Clause 9. The value of **detagged_frame_type** is as follows:

- 1) Ethernet, if the Detagged Frame uses EtherType-encapsulated IEEE 802.3 format
 - 2) RFC_1042, if the Detagged Frame is of the format specified by 10.5 in IEEE Std 802-2001 for the encoding of an IEEE 802.3 EtherType Field in an ISO/IEC 8802-2/SNAP header (this supersedes the original definition, which appeared in IETF RFC 1042 (1988))
 - 3) SNAP_Other, if the Detagged Frame contains an LLC UI PDU with DSAP and SSAP fields equal to the LLC address reserved for SNAP and the 5-octet SNAP Protocol Identifier (PID) value is not in either of the ranges used for RFC_1042 above
 - 4) LLC_Other, if the Detagged Frame contains both a DSAP and an SSAP address field in the positions specified by ISO/IEC 8802-2 Logical Link Control, but is not any of the formats described for LLC frames above
- b) Else the parameter is not present.



NOTE—The PID shown in this figure is a Protocol Identifier, as defined in 5.3 of IEEE Std 802. It is a 5-octet value, consisting of a 3-octet OUI value followed by a 2-octet locally administered identifier.

Figure 6-6—Example of operation of Port-and-Protocol-based classification

The **EtherType** value is present if the `detagged_frame_type` parameter is present and has the value `Ethernet` or `RFC_1042`. Its value is the value of the IEEE 802.3 Length/Type Field present in the Detagged Frame. The value is determined as follows:

- c) If the `detagged_frame_type` parameter is present and has the value `Ethernet` or `RFC_1042`, then this parameter is present and has the value of the IEEE 802.3 EtherType Field present in the Detagged Frame.
- d) Else the parameter is not present.

The **llc_saps** parameter is present if the `detagged_frame_type` parameter is present and has the value `LLC_Other`. Its value is determined as follows:

- e) If the detagged_frame_type parameter is present and has the value LLC_Other, then this parameter is present and its value is the pair of LLC ISO/IEC 8802-2 DSAP and SSAP address field values.
- f) Else the parameter is not present.

NOTE 1—A frame that is encapsulated using values of hex FF/FF in the position where an LLC header is to be expected (as defined by ISO/IEC 8802-2) is known as a “Novell IPX Raw” encapsulation. Such frames do not conform to ISO/IEC 8802-2, in that they do not include some of the other required LLC fields. For the purposes of this standard, they are treated as LLC_Other, regardless of whether they are legal LLC frames.

NOTE 2—Bridges are not required, for the purposes of this standard, to completely verify the format of frames as meeting ISO/IEC 8802-2 or not. They are only required to recognize the DSAP and SSAP fields of such frames.

The **snap_pid** parameter is present if the detagged_frame_type parameter is present and has the value SNAP_Other. Its value is determined as follows:

- g) If the detagged_frame_type parameter is present and has the value SNAP_Other, then the parameter is present and its value is the contents of the 5 octets following the LLC header, i.e., the PID field.
- h) Else the parameter is not present.

6.12.1 Protocol Templates

In a Bridge that supports Port-and-Protocol-based VLAN classification, a Protocol Template is a tuple that specifies a protocol to be identified in received frames. A Protocol Template has one of the following formats:

- a) A value “Ethernet” and a 16-bit IEEE 802.3 EtherType Field value
- b) A value “RFC_1042” and a 16-bit IEEE 802.3 EtherType Field value
- c) A value “SNAP_Other” and a 40-bit PID value
- d) A value “LLC_Other” and a pair of ISO/IEC 8802-2 LSAP values: DSAP and SSAP

A Protocol Template *matches* a frame if

- e) The frame’s detagged_frame_type is Ethernet, the Protocol Template is of type Ethernet, and the frame’s IEEE 802.3 EtherType Field is equal to the value of the IEEE 802.3 EtherType Field of the Protocol Template, or
- f) The frame’s detagged_frame_type is RFC_1042, the Protocol Template is of type RFC_1042 and the frame’s IEEE 802.3 EtherType Field is equal to the IEEE 802.3 EtherType Field of the Protocol Template, or
- g) The frame’s detagged_frame_type is SNAP_Other, the Protocol Template is of type SNAP_Other, and the frame’s snap_pid is equal to the PID of the Protocol Template, or
- h) The frame’s detagged_frame_type is LLC_Other, the Protocol Template is of type LLC_Other, and the frame’s llc_saps matches the value of the DSAP and SSAP of the Protocol Template.

NOTE—If a port does not support Protocol Templates of the frame’s detagged_frame_type, then no match will occur.

6.12.2 Protocol Group Identifiers

A Bridge that supports Port-and-Protocol-based VLAN classification shall support Protocol Group Identifiers.

A Protocol Group Identifier, shown as “Group Id” in Figure 6-6, designates a group of protocols that will be associated with one member of the VID Set of a Port. The association of protocols into groups is established by the contents of the Protocol Group Database, as described in 6.12.3. The identifier has scope only within a single bridge.

An implicit Protocol Group Identifier is assigned to frames that match none of the entries in the Protocol Group Database. Therefore, every incoming frame can be assigned to a Protocol Group Identifier.

6.12.3 Protocol Group Database

A Bridge that supports Port-and-Protocol-based VLAN classification shall support a single Protocol Group Database. The Protocol Group Database groups together a set of one or more Protocols by assigning them the same Protocol Group Identifier (6.12.2). Each entry of the Protocol Group Database comprises the following:

- a) A Protocol Template
- b) A Protocol Group Identifier

The Protocol Group Database specifies a mapping from Protocol Templates to Protocol Group Identifiers. If two entries of the Protocol Group Database contain different Protocol Group Identifiers, then their Protocol Templates must also be different.

The entries of the Protocol Group Database may be configured by management. A Bridge that supports Port-and-Protocol-based VLAN classification shall support at least one of the Protocol Template formats.

An implicit Protocol Group Database entry exists that matches all frames. This entry is invoked for frames that do not match the template of any of the other entries. It references an implicit Protocol Group Identifier that selects the PVID on each port. In this way, it is ensured that all incoming frames are matched by a Protocol Group Identifier and, hence, are assigned to a VID.

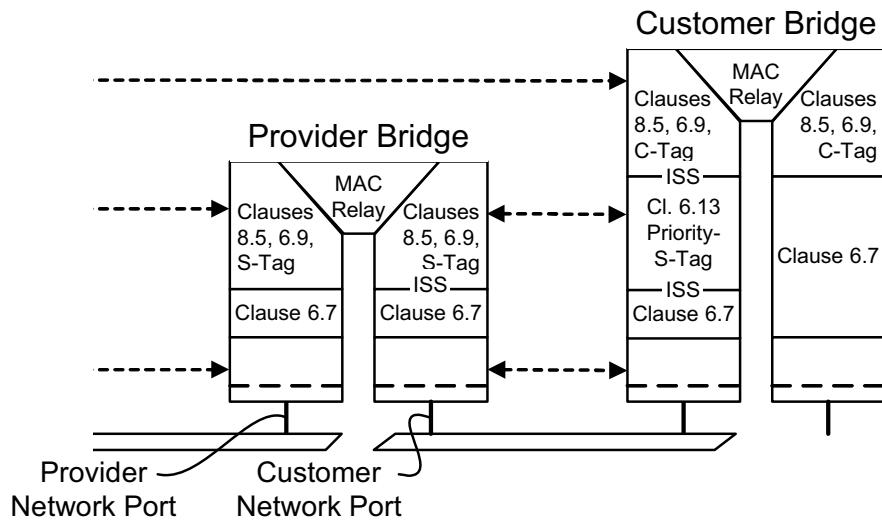
NOTE—If there are no entries in the Protocol Group Database, then the frame relay behavior of this Bridge is identical to the frame relay behavior of a Bridge having the same number of Ports that supports only Port-based VLAN classification.

6.13 Support of the ISS for attachment to a Provider Bridged Network

This standard specifies both Customer Bridges (3.25) and Provider Bridges (3.153). The operation of Provider Bridges and Provider Bridged Networks is, by design, largely transparent to Customer Bridges and Customer Bridged Local Area Networks. Figure 15-1 illustrates the relationship of the service provided by the MAC Sublayer functionality of Provider Bridges to that used by a Customer Bridge. This subclause enables a mechanism for a Customer Bridge attached to a Provider Bridged Network to request priority handling of frames.

The functions specified in this subclause provide the ISS to the MAC Relay Entity of a Customer Bridge by making use of the underlying ISS provided by the Media Access Method Convergence function as shown in Figure 6-7. These functions shall be one of the following:

- a) When Service Access Priority Selection is disabled, the functions shall be null; i.e., each M_UNITDATA.request received from the provided ISS results in an M_UNITDATA.request with identical parameters presented to the underlying ISS, and each M_UNITDATA.indication received from the underlying ISS results in an M_UNITDATA.indication with identical parameters presented to the provided ISS.
- b) When Service Access Priority Selection is enabled, the mac_service_data_unit in each M_UNIDATA.request is priority-tagged with an S-VLAN tag header when presented to the underlying ISS, and if an S-VLAN tag header is present in the mac_service_data_unit parameter in an M_UNITDATA.indication received from the underlying ISS, the S-VLAN tag header is removed before the M_UNITDATA.indication is presented to the provided ISS.

**Figure 6-7—Service access priority selection**

Specification of a null function allows a Customer Bridge without additional functionality to connect to a Provider Bridged Network.

Specification of the Service Access Priority Selection function allows the Customer Bridge Port to request priority handling of the frame from a Port-based service interface (15.3) to a Provider Bridged Network on the basis of the priority assigned within the Customer Bridged Local Area Network, while allowing for differences in cost and service level ascribed to individual values of priority within the service provider and customer networks.

6.13.1 Data requests

For each M_UNITDATA.request received from the provided ISS, an M_UNITDATA.request is presented to the underlying ISS with the following parameters:

The destination_address, source_address, and priority parameters are unchanged.

A tag header, formatted as necessary for the destination MAC type, is inserted as the initial octets of the mac_service_data_unit parameter. The Tag Protocol Identification is the Service VLAN Tag EtherType (9.5). The Tag Control Information contains a null VLAN Identifier field, and PCP and DEI fields constructed as specified in 6.9.3 using a drop_eligible value of false and a service_access_priority value derived from the priority parameter using the Service Access Priority Table 6-7.

Table 6-7 specifies default service_access_priority values for each of the eight possible values of the priority parameter in the M_UNITDATA.request received from the provided ISS. These default values shall be used as the initial values of the corresponding entries of the Service Access Priority Table for each Port supporting Service Access Priority Selection. The values in the table may be modified by management. If this capability is provided, the value of the table entries shall be independently settable for each Port and for each value of priority, and the Bridge shall have the capability to use the full range of the values in the parameter ranges specified in the table.

The frame_check_sequence parameter is either unspecified or contains a value modified from the received frame_check_sequence value taking into account the changes in the mac_service_data_unit parameter.

Table 6-7—Service Access Priority

Received priority	Default service access priority	Range
0	0	0–7
1	1	0–7
2	2	0–7
3	3	0–7
4	4	0–7
5	5	0–7
6	6	0–7
7	7	0–7

NOTE—The priority of the original service request made of the customer network can be carried transparently and unchanged across the Provider Bridged Network in the Customer VLAN tag.

6.13.2 Data indications

For each M_UNITDATA.indication received from the underlying ISS, an M_UNITDATA.request is presented to the supported ISS with the following parameters:

The destination_address and source_address parameters are unchanged.

If the initial octets of the mac_service_data_unit do not contain a tag header with the Service VLAN tag EtherType value (9.5), then the mac_service_data_unit and frame_check_sequence parameters are unchanged; otherwise,

- a) the mac_service_data_unit is detagged by removing the octets of the Service VLAN tag header, and
- b) the priority parameter is decoded from the Priority Code Point field (6.9.3), and
- c) the frame_check_sequence parameter is either unspecified or contains a value modified from the received frame_check_sequence value taking into account the changes in the mac_service_data_unit parameter.

6.14 Support of the ISS within a system

A single system can comprise two or more instances of VLAN-aware Bridge components connected by one or more of their Ports. If those Ports are not otherwise accessible, the ISS may be provided by means outside the scope of this specification. Figure 15-4 provides an example. Each instance of such an implementation of the ISS shall support the MAC status and Point-to-point parameters. An M_UNITDATA.request at one of the Ports connected to such an instance of the ISS shall result in M_UNITDATA.indications with identical parameters at all other Ports connected to that instance.

For convenience of management, such instances of ISS provision are assigned the LAN MAC Type 802.1. Management parameters common to MACs of this type are specified in Clause 12.

NOTE—The ISS can also be supported at Bridge Ports wholly contained within a system by any IEEE 802 LAN technology, subject to the system requirements of the particular media access method.

6.15 Support of the ISS by additional technologies

The ISS may be supported by other technologies that provide either an IEEE 802 MAC Service or an emulated IEEE 802 MAC Service. The technology is responsible for invoking an `M_UNIDATA.indication` with appropriate parameters (6.6) for each received frame, and transmitting a frame in response to each `M_UNITDATA.request`. The `mac_service_data_unit` parameter in an `M_UNITDATA.indication` may include a tag header of a type recognized by the functions using the EISS (6.8). If multiplexing of multiple virtual instances of the MAC Service is provided, each virtual service must be

- a) identified by the VID field in a tag header of a type recognized by the functions using the EISS, or
- b) attached to separate instance of the ISS with no tag header in the `mac_service_data_unit`

(i.e., the only multiplexing recognized by the functions of the MAC Relay are those identified by a VID).

6.16 Filtering services in Bridged Local Area Networks

MAC Bridges provide filtering services that support aspects of the maintenance of QoS—in particular, transit delay, priority, and throughput. In addition, these services provide a degree of administrative control over the propagation of particular MAC Addresses in the network.

The services described are services in the most general sense; i.e., descriptions of functionality available to a MAC Service user or an administrator to control and access filtering capabilities. The descriptions make no assumptions as to how the service might be realized. There are at least the following possibilities:

- a) Use of existing protocols and mechanisms, defined in IEEE 802[®] standards and elsewhere.
- b) Use of management functionality, either locally defined or via remote management protocols.
- c) Other means, standardized or otherwise.

6.16.1 Purpose(s) of filtering service provision

Filtering services are provided for the purposes described in 6.16.1.1 and 6.16.1.2.

6.16.1.1 Administrative control

Filtering services provide administrative control over the use of particular source and destination addresses in designated parts of the network. Such control allows network managers and administrators to limit the extent of operation of network layer and other protocols that use individual and group MAC addresses by establishing administrative boundaries across which specific MAC Addresses are not forwarded.

6.16.1.2 Throughput and end station load

Filtering services increase the overall throughput of the network, and reduce the load placed on end stations caused by the reception of frames that are destined for other end stations, by

- a) Limiting frames destined for specific MAC Addresses to parts of the network that, to a high probability, lie along a path between the source MAC Address and the destination MAC Address.
- b) Reducing the extent of group addressed frames to those parts of the network that contain end stations that are legitimate recipients of that traffic.
- c) In Virtual Bridged Local Area Networks, reducing the extent of frames on specific VLANs to those parts of the network that contain stations that are legitimate recipients of traffic on that VLAN.

NOTE—Some aspects of the filtering services described in this standard are dependent upon the active participation of end stations. Where such participation is not possible, those aspects of the filtering services will be unavailable.

6.16.2 Goals of filtering service provision

The filtering services provided can be used to

- a) Allow the MAC Service provider to dynamically learn where the recipients of frames addressed to individual MAC Addresses are located.
- b) Allow end stations that are the potential recipients of MAC frames destined for group MAC addresses to dynamically indicate to the MAC Service provider which destination MAC Address(es) they wish to receive.
- c) Exercise administrative control over the extent of propagation of specific MAC Addresses.

6.16.3 Users of filtering services

The filtering services provided are available to the following users:

- a) Network management and administration, for the purposes of applying administrative control. Interactions between administrators of the network and the filtering service provider may be achieved by local means or by means of explicit management mechanisms.
- b) End stations, for the purposes of controlling the destination addresses that they will receive. Interactions between end stations and the filtering service provider may be implicit, as is the case with the Learning Process (8.7), or by explicit use of filtering service primitives.

6.16.4 Basis of service

Filtering in Bridged Local Area Networks relies on the establishment of filtering rules, and subsequent filtering decisions, that are based on the value(s) contained in the Source MAC Address, Destination MAC Address, or VID fields in MAC frames.

NOTE—The filtering services defined by this standard use source address learning, destination address, and VID filtering.

6.16.5 Categories of service

Filtering services in Bridged Local Area Networks fall into the following categories:

- a) *Basic Filtering Services.* These services are supported by the Forwarding Process (8.6) and by Static Filtering Entries (8.8.1), Static VLAN Registration Entries (8.8.2), Dynamic Filtering Entries (8.8.3), and Dynamic VLAN Registration Entries (8.8.5) in the Filtering Database. The information contained in the Dynamic Filtering Entries is maintained through the operation of the Learning Process (8.7), while the information contained in the Dynamic VLAN Registration Entries is maintained through the operation of MVRP (11.2).
- b) *Extended Filtering Services.* These services are supported by the Forwarding Process (8.6), and the Static Filtering Entries (8.8.1) and MAC Address Registration Entries (8.8.4) in the Filtering Database. The information contained in the MAC Address Registration Entries is maintained through the operation of MMRP (10.9).

All Bridges shall support Basic Filtering Services and may support Extended Filtering Services.

6.16.6 Service configuration

In the absence of explicit information in the Filtering Database, the behavior of the Forwarding Process with respect to the forwarding or filtering of frames destined for group MAC addresses depends upon the categories of service supported by the Bridge.

Basic Filtering Services support the filtering behavior required for regions of a Bridged Local Area Network in which potential recipients of multicast frames exist, but where either the recipients or the Bridges are unable to support the dynamic configuration of filtering information for those group MAC addresses, or the recipients have a requirement to receive all traffic destined for group MAC addresses.

Extended Filtering Services support the filtering behavior required for regions of a network in which potential recipients of multicast frames exist, and where both the potential recipients of frames and the Bridges are able to support dynamic configuration of filtering information for group MAC addresses. In order to integrate this extended filtering behavior with the needs of regions of the network that support only Basic Filtering Services, Bridges that support Extended Filtering Services can be statically and dynamically configured to modify their filtering behavior on a per-group MAC address basis, and also on the basis of the overall filtering service provided by each outbound Port with regard to multicast frames. The latter capability permits configuration of the Port's default forwarding or filtering behavior with regard to group MAC addresses for which no specific static or dynamic filtering information has been configured.

Service configuration provides the ability to configure the overall filtering for the following cases:

- a) Bridges that only implement Basic Filtering Services.
- b) Bridges that support Extended Filtering Services in a heterogeneous environment, where some equipment is unable to participate in Dynamic Multicast Filtering, or where some equipment (e.g., routers) has specific needs to see unfiltered traffic.
- c) Bridges that support Extended Filtering Services in a homogeneous environment, where all equipment is able to participate in Dynamic Multicast Filtering.

6.16.7 Service definition for Extended Filtering Services

The Filtering Services are described by means of service primitives that define particular types of interaction between MAC Service users and the MAC Service provider across the MAC Service boundary. As these interactions are not defined between peer entities, they are described simply in terms of service requests sent from the MAC Service user to the MAC Service provider.

6.16.7.1 Dynamic registration and de-registration services

These services allow MAC Service users dynamic control over the set of destination MAC Addresses that they will receive from the MAC Service provider, by:

- a) Registering/deregistering membership of specific Groups associated with those addresses.
- b) Registering/deregistering individual MAC address information.
- c) Registering/deregistering service requirements with regard to the overall forwarding/filtering behavior for Groups.

Provision of these services is achieved by means of MMRP and its associated procedures, as described in 10.9.

NOTE 1—These services can provide the MAC Service user with dynamic control over access to multicast data streams, for example, multiple video channels made available by a server using a different group MAC address for each channel. The ability to both register and deregister Group membership, coupled with the filtering action associated with the

Group membership, limits the impact of such services on the bandwidth available in the network. These services can be used to control the reception of other categories of multicast traffic, for similar reasons.

REGISTER_MAC_ADDRESS (MAC_ADDRESS)

Indicates to the MAC Service provider that the MAC Service user wishes to receive frames containing the MAC Address indicated in the MAC_ADDRESS parameter as the destination address. The MAC Addresses that can be carried by this parameter do not include the following:

- d) Any of the Reserved Addresses identified in Table 8-1, Table 8-2, or Table 8-3.
- e) Any of the MRP Application addresses, as defined in Table 10-1.

Deregister_MAC_Address (MAC_ADDRESS)

Indicates to the MAC Service provider that the MAC Service user no longer wishes to receive frames containing the MAC Address indicated in the MAC_ADDRESS parameter as the destination address.

REGISTER_SERVICE_REQUIREMENT (REQUIREMENT_SPECIFICATION)

Indicates to the MAC Service provider that the MAC Service user has a requirement for any devices that support Extended Filtering Services to forward frames in the direction of the MAC Service user in accordance with the definition of the service requirement defined by the REQUIREMENT_SPECIFICATION parameter. The values that can be carried by this parameter are as follows:

- f) Forward All Groups.
- g) Forward Unregistered Groups.

Deregister_Service_Requirement (REQUIREMENT_SPECIFICATION)

Indicates to the MAC Service provider that the MAC Service user no longer has a requirement for any devices that support Extended Filtering Services to forward frames in the direction of the MAC Service user in accordance with the definition of the service requirement defined by the REQUIREMENT_SPECIFICATION parameter. The values that can be carried by this parameter are as follows:

- h) Forward All Groups.
- i) Forward Unregistered Groups.

The use of these services can result in the propagation of group MAC address and service requirement information across the Spanning Tree, affecting the contents of MAC Address Registration Entries (8.8.4) in Bridges and end stations, and thereby affecting the frame forwarding behavior of the Bridges and end stations with regard to multicast frames.

NOTE 2—If the Enhanced Internal Sublayer Service (EISS) is supported, the Extended Filtering Service primitives issued by the MAC Service user should also include a VID parameter in order to identify the VLAN associated with the MAC_ADDRESS or REQUIREMENT_SPECIFICATION specified.

6.17 EISS Multiplex Entity

The EISS Multiplex Entity enables shims defined for the ISS to use the EISS. Figure 6-8 illustrates two EISS Multiplex Entities placed back-to-back.

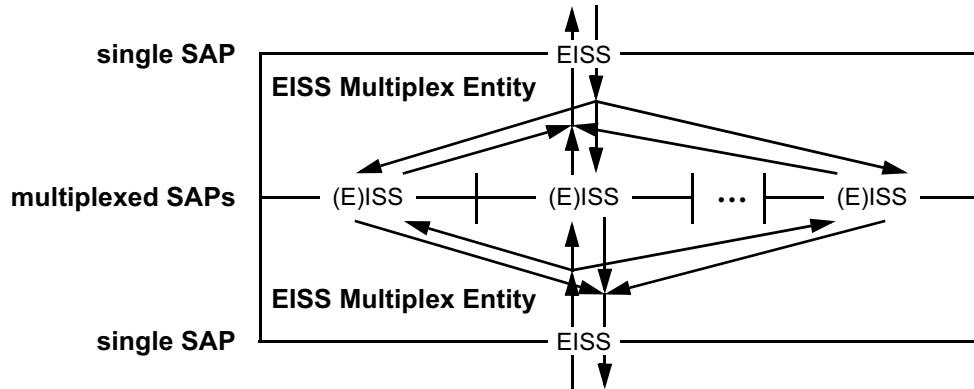


Figure 6-8—Two back-to-back EISS Multiplex Entities

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 multiplexed 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 6-8 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 6-8 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.

6.18 Backbone Service Instance Multiplex Entity

The Backbone Service Instance Multiplex Entity allows shims defined for the ISS to be instantiated per backbone service instance at a Service Access Point that supports multiple backbone service instances. Figure 6-9 illustrates two Backbone Service Instance Multiplex Entities placed back-to-back. A set of back-to-back Backbone Service Instance Multiplex Entities may be used at the ISS of a Provider Instance Port or Customer Backbone Port to provide per Backbone Service Instance SAPs that support Connectivity Fault Management (CFM) shims at the Backbone Service Instance MD Level as shown in Figure 6-10 and the figures of 26.8.

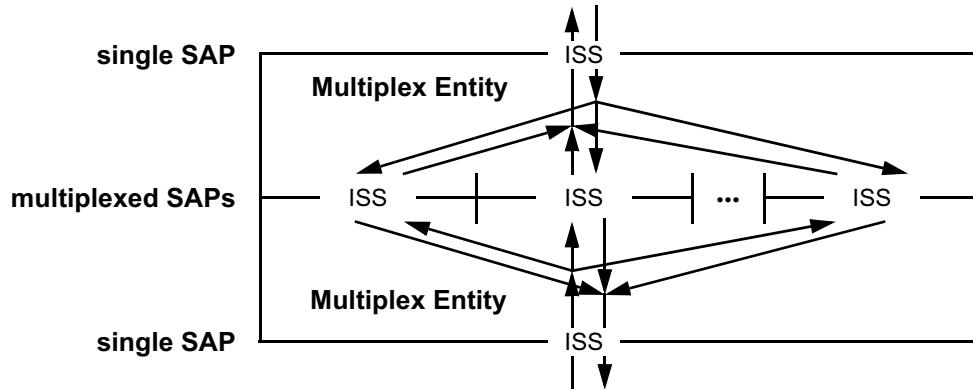


Figure 6-9—Two back-to-back Backbone Service Instance Multiplex Entities

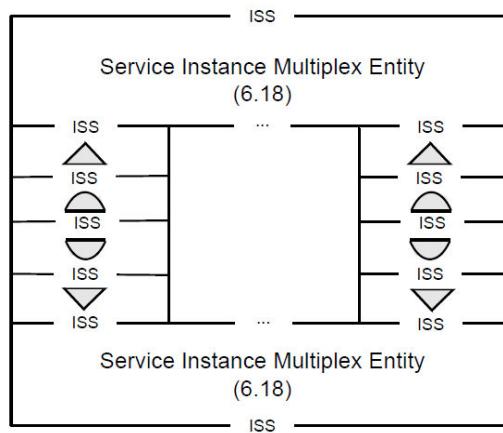


Figure 6-10—Backbone Service Instance Multiplex Entities with example CFM shims

NOTE 1—Figure 6-10 shows a representative placement of CFM shims between back-to-back Backbone Service Instance Multiplex Entities. This is not intended to imply any restriction that this is the only possible configuration of MEPs and MIPs.

A Backbone Service Instance Multiplex Entity has one ISS SAP that supports multiple backbone service instances, and a number of multiplexed ISS SAPs each supporting a single backbone service instance. Each multiplexed SAP has a single assigned I-SID value. Every I-SID value is assigned to a multiplexed SAP, and no I-SID value is assigned to more than one multiplexed SAP.

NOTE 2—There are implicitly 2^{24} multiplexed SAPs—one for each potential I-SID value. This does not require any implementation to support 2^{24} SAPs. The combined structure of back-to-back Backbone Service Instance Multiplex Entities is transparent to any frames with an I-SID value corresponding to a multiplexed SAP that does not have a protocol entity (e.g., CFM) instantiated at that SAP.

In the multiplexing direction, any Request or Indication received from a multiplexed SAP results in the generation of a corresponding Request or Indication to the single ISS SAP, with a Backbone Service Instance tag header (I-TAG) added to the mac_service_data_unit as specified in 6.18.2. Similarly, in the demultiplexing direction any Request or Indication received from the single ISS SAP results in the generation of a corresponding Request or Indication at one of the multiplexed SAPs (determined by the I-SID contained in a I-TAG in the mac_service_data_unit), with the I-TAG removed from the mac_service_data_unit as specified in 6.18.1. When a Backbone Service Instance tag header with the UCA field containing a value of zero is removed, the C_DA and C_SA fields of the I-TAG remain in the mac_service_data_unit. In this case, a new EtherType field is prepended so the mac_service_data_unit still begins with a valid EtherType value. The Encapsulated Addresses EtherType value in Table 6-8 is allocated for this purpose.

Table 6-8—Encapsulated Addresses EtherType

Name	Value
IEEE 802.1Q Encapsulated Addresses EtherType	89-10

NOTE 3—This EtherType value is used only internally at the Backbone Service Multiplex Entity and does not appear in any frame on a LAN in the PBBN.

6.18.1 Demultiplexing direction

On receipt of an M_UNITDATA.request or M_UNITDATA.indication primitive from the single ISS SAP, the received frame shall be discarded if

- a) The initial octets of the mac_service_data_unit do not contain a valid I-TAG; or
- b) The Res2 field in the I-TAG is not zero.

Otherwise, an M_UNITDATA.request or M_UNITDATA.indication primitive is invoked at the multiplexed ISS SAP corresponding to the I-SID value in the I-TAG. The parameter values are determined as follows:

The **connection_identifier** parameter contains the same value as in the received primitive.

The **destination_address** and **source_address** parameters are determined as follows:

- c) If the UCA bit is zero, the **destination_address** and **source_address** parameters contain the respective values in the received primitive;
- d) Otherwise, the **destination_address** and **source_address** parameters contain the respective values in the C-DA and C-SA fields of the I-TAG.

The value of the **drop_eligible** and **priority** parameters are decoded from the I-DEI and the I-PCP fields of the I-TAG header in the mac_service_data_unit as described in 6.18.3.

The value of the **mac_service_data_unit** parameter is constructed from the value of the received mac_service_data_unit modified as follows:

- e) If the UCA field in the I-TAG is zero, then the I-PCP, I-DEI, UCA, Res1, Res2, and I-SID fields of the I-TAG are removed and the Backbone Service Instance tag EtherType value is replaced with the Encapsulated Addresses EtherType value (Table 6-8);
- f) Otherwise, the entire I-TAG is removed.

The value of the **frame_check_sequence** parameter is either unspecified or derived from the received FCS information modified as necessary to take into account changes to the frame.

6.18.2 Multiplexing direction

On receipt of an M_UNITDATA.request or M_UNITDATA.indication primitive from any of the multiplexed ISS SAPs, an M_UNITDATA.request or M_UNITDATA.indication primitive is invoked at the single ISS SAP. The parameter values are determined as follows:

The **source_address**, **priority**, **drop_eligible** and **connection_identifier** parameters contain the same value as in the received primitive.

The **destination_address** parameter is determined as follows:

- a) If the destination_address in the received primitive is a group address and the UCA bit is set, then the **destination_address** parameter contains the Backbone Service Instance Group address constructed from the Backbone Service Instance Identifier corresponding to the multiplexed ISS SAP at which the primitive was received;
- b) Otherwise the **destination_address** parameter contains the same value as in the received primitive.

The value of the **mac_service_data_unit** parameter is equal to the value of the received mac_service_data_unit modified as follows:

- c) If the initial two octets of the mac_service_data_unit match the Encapsulated Addresses EtherType value, then these octets are replaced with the a Backbone Service Instance tag EtherType value, and the I-PCP, I-DEI, UCA, Res1, Res2, and I-SID fields of an I-TAG are inserted following this EtherType value. The UCA, Res1 and Res2 fields all contain a value of zero. The I-SID field contains the value of the Backbone Service Instance Identifier corresponding to the multiplexed ISS SAP at which the primitive was received. The values of the I-PCP and I-DEI fields are encoded from the **drop_eligible** and **priority** parameters as specified in 6.18.3.
- d) If the initial two octets of the mac_service_data_unit do not match the Encapsulated Addresses EtherType value, then a complete I-TAG is prepended to the mac_service_data_unit. The UCA field contains a value of one. The Res1 and Res2 fields contain a value of zero. The C-DA and C-SA fields contain the destination_address and source_address from the received primitive, respectively. The I-SID field contains the value of the Backbone Service Instance Identifier corresponding to the multiplexed ISS SAP at which the primitive was received. The values of the I-PCP and I-DEI fields are encoded from the **drop_eligible** and **priority** parameters as specified in 6.18.3.

The value of the **frame_check_sequence** parameter is either unspecified or derived from the received FCS information modified as necessary to take into account changes to the frame.

6.18.3 Priority Code Point encoding

The encoding (and decoding) of the I-PCP and I-DEI fields from (to) the **priority** and **drop_eligible** parameters is accomplished using a Priority Code Point Encoding table (and Priority Code Point Decoding Table) as described in 6.9.3.

6.18.4 Status parameters

The MAC_Operational status parameter (6.6.2) presented to the uppermost single ISS SAP in Figure 6-9 is TRUE if and only if

- a) The uppermost single ISS 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 multiplexed SAPs in Figure 6-9 is computed separately, and is TRUE if and only if

- c) The lowermost single ISS SAP's MAC_Operational parameter is TRUE; and
- d) That multiplexed SAP's MAC_Enabled parameter is TRUE.

6.19 TESI Multiplex Entity

The TESI Multiplex Entity allows shims defined for PBB-TE to be instantiated per TESI at a Service Access Point that supports multiple TESIs. Figure 6-11 illustrates two TESI Multiplex Entities, placed back-to-back. A set of back-to-back TESI Multiplex Entities may be used at the multiplexed (E)ISS SAPs that are associated with ESP-VIDs on a Customer Backbone Port to provide per TESI SAPs that support Connectivity Fault Management (CFM) shims for PBB-TE MAs as shown in Figure 26-8.

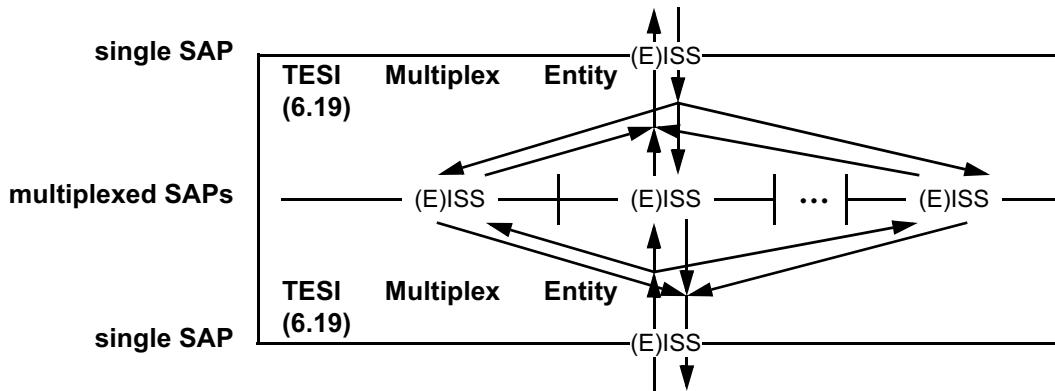


Figure 6-11—Two back-to-back Up and Down TESI Multiplex Entities

A TESI Multiplex Entity has one (E)ISS SAP that supports multiple TESIs, and a number of multiplexed (E)ISS SAPs each supporting a single TESI. Each multiplexed SAP has destination_address, source_address, and vlan_identifier combinations assigned by the TESI Multiplex Entity. Every destination_address, source_address, and vlan_identifier combination can be assigned to a multiplexed SAP, and no destination_address, source_address, and vlan_identifier combination is assigned to more than one multiplexed SAP.

Upon receiving a Request or Indication from its single (E)ISS SAP, the TESI Multiplex Entity uses the destination_address, source_address, and vlan_identifier to select the corresponding one of its multiplexed SAPs to present the Request or Indication. The Request or Indication presented at the multiplexed SAPs has the same parameters as the original Request or Indication at the Single SAP. Similarly, any Request or Indication received from a multiplexed SAP is transparently presented to the single (E)ISS SAP.

The MAC_Operational status parameter (6.8.2) presented to the uppermost single (E)ISS SAP in Figure 6-11 is TRUE if and only if

- a) The uppermost single (E)ISS 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 multiplexed SAPs in Figure 6-11 is computed separately, and is TRUE if and only if

- c) The lowermost single (E)ISS SAP's MAC_Operational parameter is TRUE; and
- d) That multiplexed SAP's MAC_Enabled parameter is TRUE.

The MAC_Operational parameter of the passive SAP on a PBB-TE MEP is set to FALSE when the PBB-TE MEP declares an errorCCMdefect (20.21.3) or an xconCCMdefect (20.23.3), setting the MAC_Operational parameter of the associated multiplexed SAP on the TESI Multiplex Entity to FALSE. The MAC_Operational parameter of the passive SAP on a PBB-TE MEP is set back to TRUE when both defects are cleared, setting back to TRUE the MAC_Operational parameter of the associated multiplexed SAP.

6.20 Support of the ISS with signaled priority

The functions specified in this subclause comprise a shim (3.168, 6.1.5) that uses an ISS SAP supported by a MAC entity (6.7) to provide explicitly signaled priority information to an ISS SAP that supports a VLAN-unaware MAC Relay (Figure 6-12). Signaled priority information is derived from a VLAN tag header (9.3), if such header is present in the mac_service_data_unit. Any given instance of the ISS shall be supported by recognizing one but not both of the following VLAN tag EtherTypes:

- a) Customer VLAN tag (C-TAG); or
- b) Service VLAN tag (S-TAG).

The VLAN tag EtherType is selected as specified in 9.5.

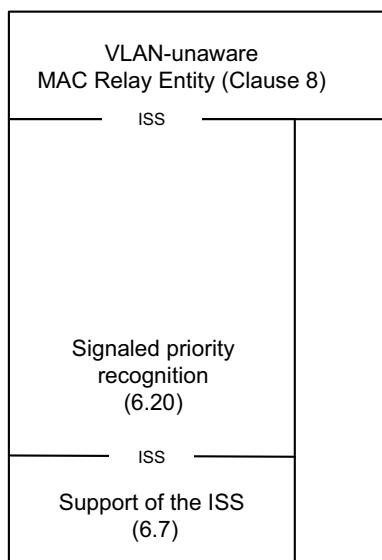


Figure 6-12—Supporting the ISS with signaled priority

6.20.1 Data indications

On receipt of an M_UNITDATA.indication primitive from the lower ISS SAP (Figure 6-12), an M_UNITDATA.indication primitive is invoked at the upper ISS SAP, with parameter values determined as follows:

The **destination_address**, **source_address**, **connection_identifier**, **mac_service_data_unit**, and **frame_check_sequence** parameters carry values equal to the corresponding parameters in the received data indication.

The value of the **drop_eligible** and **priority** parameters are determined as follows:

- a) If the initial octets of the mac_service_data_unit contain a valid VLAN tag header (9.3) of the type used to support the EISS (9.5), the value of the drop_eligible parameter and the received priority value are decoded from the tag header as described in 6.9.3. Otherwise;
- b) The received priority value and the drop_eligible parameter value are the values in the received data indication.
- c) The value of the priority parameter is then regenerated from the received priority, as specified in 6.9.4.

6.20.2 Data requests

On receipt of an M_UNITDATA.request primitive from the upper ISS SAP (Figure 6-12), an M-UNITDATA.request primitive with identical parameter values is invoked at the lower ISS SAP.

7. Principles of network operation

This clause establishes the principles and a model of Virtual Bridged Local Area Network operation. It defines the context necessary for:

- a) The operation of individual VLAN-aware Bridges (Clause 8);
- b) Their participation in the Spanning Tree (Clause 8 of IEEE Std 802.1D, 1998 Edition), Rapid Spanning Tree (Clause 13), or Multiple Spanning Tree Protocol (Clause 13);
- c) The management of individual Bridges (Clause 12); and
- d) The management of VLAN Topology (Clause 11)

to support, preserve, and maintain the quality of the MAC Service as discussed in Clause 6.

7.1 Network overview

The operation of a Virtual Bridged Local Area Network, the Bridges, and the LANs, that compose that network comprises:

- a) A physical topology comprising LANs, Bridges, and Bridge Ports. Each Bridge Port attaches a LAN to a Bridge and is capable of providing bidirectional connectivity for MAC user data frames. Each LAN is connected to every other LAN by a Bridge and zero or more other LANs and Bridges.
- b) Calculation of one or more active topologies, each a loop-free subset of the physical topology.
- c) Rules for the classification of MAC user data frames that allow each Bridge to allocate, directly or indirectly, each frame to one and only one active topology.
- d) Management control of the connectivity provided for differently classified data frames by the selected active topology.
- e) Implicit or explicit configuration of end station location information, identifying LANs with attached end stations that need to receive user data frames with a given destination address.
- f) Communication of end station location information to allow Bridges to restrict user data frames to LANs in the path provided to their destination(s) by the chosen active topology.

These elements and their interrelationships are illustrated in Figure 7-1.

NOTE 1—The physical and active topologies can be represented as bi-partite graphs. Bridges and LANs are nodes in these graphs (including LANs that are point-to-point links) and the Bridge Ports are edges.

NOTE 2—This standard applies the notion of a physical topology to media access control methods, like wireless, where there is no tangible physical connection between a Bridge and an attached LAN. A Bridge Port models this association just as for wired connectivity.

NOTE 3—Each of the active topologies [see item b)] does not necessarily span the entire network, but does span all those Bridges and LANs between which data frame connectivity is desired for frames allocated to that active topology.

NOTE 4—The destination addressing information associated with a MAC user data frame includes the VLAN classification of the frame [see item d)].

NOTE 5—Implicit configuration [see item d)], or recognition, of end station location includes observation of the source address of the user data frame transmitted by that station. The user data frame communicates that location information [see item e)] along the portion of the chosen active topology leading to the frame's destination(s).

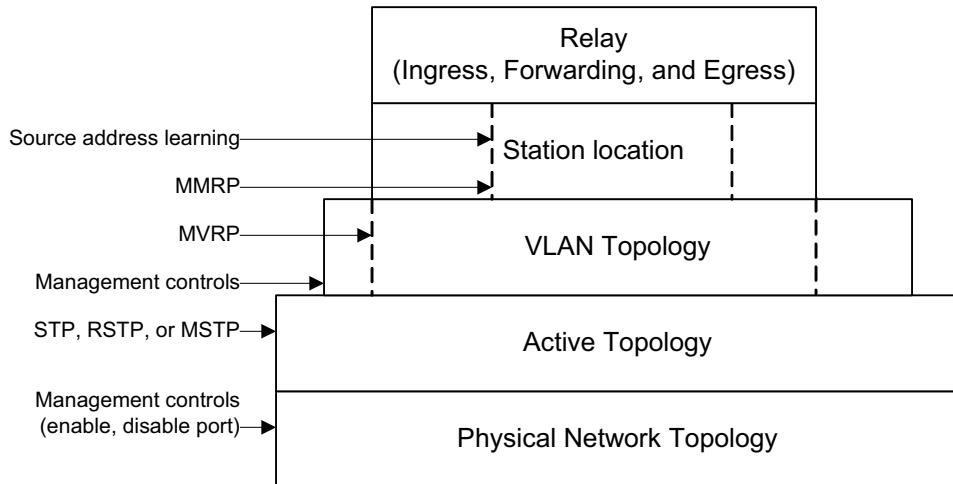


Figure 7-1—VLAN Bridging overview

7.2 Use of VLANs

Virtual Local Area Networks (VLANs) and their VLAN Identifiers (VIDs) provide a convenient and consistent network-wide reference for Bridges to:

- Identify rules for the classification of user data frames into VLANs;
- Effectively extend the source and destination MAC addresses, by treating frames and addressing information for different VLANs independently;
- Identify and select from different active topologies;
- Identify the configuration parameters that partition or restrict access from one part of the network to another.

Taken together these capabilities allow VLAN-aware Bridges to emulate a number of separately manageable, or virtual, Bridged Local Area Networks. A LAN that has been selected by network management to receive frames assigned to a given VLAN is said to form part of, belong to, or be a member of the VLAN. Similarly, end stations that are attached to those LANs and that can receive frames assigned to the VLAN are said to be attached to that VLAN.

NOTE—Separate control over the transmission and over the reception of frames to and from LANs and end stations is possible. To avoid ambiguity, VLAN membership is defined by reception.

Inclusion of the VID in VLAN-tagged frames guards against connectivity loops arising from differing classifications by different Bridges, and permits enhanced classification rules to be used by some Bridges, while others simply forward the previously classified frames.

The VLAN tag allows frames to carry priority information, even if the frame has not been classified as belonging to a particular VLAN.

7.3 VLAN topology

Each Bridge cooperates with others to operate a spanning tree protocol to calculate one or more loop-free, fully connected, active topologies. This calculation supports the quality of the MAC Service (Clause 6) and provides rapid recovery from network component failure, by using alternate physical connectivity, without requiring management intervention.

All user data frames classified as belonging to a given VLAN are constrained by the forwarding process of each Bridge to a single active topology. Each and every VLAN is thus associated with a spanning tree, although more than one VLAN can be associated with any given tree. Each VLAN may occupy the full extent of the active topology of its associated spanning tree or a connected subset of that active topology. The maximum extent of the connected subset may be bounded by management by explicitly excluding certain Bridge Ports from a VLAN's connectivity.

At any time the current extent of a VLAN can be further reduced from the maximum to include only those LANs that provide communication between attached devices, by the use of protocol that allows end stations to request and release services that use the VLAN. Such a protocol is specified in Clause 11. The dynamic determination of VLAN extent provides flexibility and bandwidth conservation, at the cost of network management complexity.

NOTE 1—Dynamic determination of VLAN extent is generally preferable to static configuration for bandwidth conservation, as the latter is error prone and can defeat potential alternate connectivity-requiring active management intervention to recover from network component failure.

NOTE 2—To accommodate end stations that do not participate in the MVRP specified in Clause 11, management controls associated with each Bridge Port allow the Port to identify the attached LAN as connecting end stations that require services using specified VLANs.

7.4 Locating end stations

Functioning as a distributed system, Bridges within the current extent of a VLAN can, through explicit and or implicit cooperation, locate those LANs where an attached end station or end stations are intended to receive frames addressed to a specified individual address or group address. Bridges can thus reduce traffic by confining frames to the LANs where their transmission is necessary.

NOTE 1—Individually Bridges do not determine the precise location of end stations but merely determine which of their Bridge Ports need to forward frames toward the destination(s). For the system of Bridges this is sufficient to restrict frames to the paths necessary to reach the destination LANs.

The multiple MAC registration protocol (MMRP), specified in Clause 10, allows end stations to advertise their presence and their desire to join (or leave) a multicast group, or to register an individual MAC address, in the context of a VLAN. The protocol communicates this information to other Bridges, using the VLAN and its active topology.

NOTE 2—To accommodate end stations that do not participate in MMRP, management controls associated with each Bridge Port allow the Port to identify the attached LANs as connecting end stations that are intended to receive specified group addresses. The continuous operation of MMRP and the propagation of location information through Bridges using the current active topology for the VLAN support multicast traffic reduction, while ensuring rapid restoration of multicast connectivity without management intervention if alternate connectivity is selected following network component failure.

Each end station implicitly advertises its attachment to a LAN and its individual MAC address whenever it transmits a frame. Bridges learn from the source address as they forward the frame along the active topology to its destination or destinations – or throughout the VLAN if the location of the destination or destinations is unknown. The learned information is stored in the Filtering Database used to filter frames on the basis of their destination addresses.

The Filtering Database architecture defined in this standard recognizes that

- a) For some configurations, it is necessary to allow address information learned in one VLAN to be shared among a number of VLANs. This is known as *Shared VLAN Learning* (3.165);
- b) For some configurations, it is desirable to ensure that address information learned in one VLAN is not shared with other VLANs. This is known as *Independent VLAN Learning* (3.75);
- c) For some configurations, it is immaterial as to whether learned information is shared between VLANs.

NOTE 1—Annex B discusses the need for Shared and Independent VLAN Learning and some related interoperability issues.

Shared VLAN Learning is achieved by including learned information from a number of VLANs in the same Filtering Database; Independent VLAN Learning is achieved by including information from each VLAN in distinct Filtering Databases.

NOTE 2—The actual Filtering Database specification specifies a single Filtering Database that, through the inclusion of VLAN identification information in each database entry, can model the existence of one or more distinct Filtering Databases.

Within a given network, there may be a combination of configuration requirements, so that individual Bridges may be called on to share learned information, or not share it, according to the requirements of particular VLANs or groups of VLANs. The Filtering Database structure that is defined in this standard allows both Shared and Independent VLAN Learning to be implemented within the same Bridge; i.e., allows learned information to be shared between those VLANs for which Shared VLAN Learning is necessary, while also allowing learned information not to be shared between those VLANs for which Independent VLAN Learning is necessary. The precise requirements for each VLAN with respect to sharing or independence of learned information (if any) are made known to Bridges by means of a set of *VLAN Learning Constraints* (8.8.8.2) and fixed allocations of VIDs to filtering databases (8.8.8.1), which may be configured into the Bridges by means of management operations. By analyzing the set of learning constraints and fixed allocations for the VIDs that are currently active, the Bridge can determine

- d) How many independent Filtering Databases are required in order to meet the constraints;
- e) For each VID, which Filtering Database it will feed any learned information into (and use learned information from).

The manner in which this mapping of VIDs onto Filtering Databases is achieved is defined in 8.8.8; the result is that each VID is associated with exactly one Filtering Database.

The most general application of the Filtering Database specification in this standard is a Bridge that can support M independent Filtering Databases and can map N VIDs onto each Filtering Database. Such a Bridge is known as an SVL/IVL Bridge (3.167).

The conformance requirements in this standard (5.4) recognize that Bridges will be implemented with differing capabilities in order to meet a wide range of application needs, and that the full generality of the SVL/IVL approach is not always either necessary or desirable, as observed in the discussion in Annex F. In a given conformant implementation, there may be restrictions placed on the number of Filtering Databases that can be supported and/or the number of VIDs that can be mapped onto each Filtering Database. The full spectrum of conformant Filtering Database implementations is therefore as follows:

- f) The SVL/IVL Bridge, as described above. Such Bridges provide support for M Filtering Databases, with the ability to map N VIDs onto each one;
- g) Support for a single Filtering Database only. MAC Address information that is learned in association with one VID can be used in filtering decisions taken relative to all other VIDs supported by the Bridge. Bridges that support a single Filtering Database are referred to as SVL Bridges;

- h) Support for multiple Filtering Databases, but only a single VID can be mapped onto each Filtering Database. MAC Address information that is learned in association with one VID cannot be used in filtering decisions taken relative to any other VID. Bridges that support this mode of operation are referred to as IVL Bridges.

Clause 15 and Clause 16 describe how the mapping of Customer VLANs (C-VLANs) into Service VLANs (S-VLANs) is accomplished within Provider Bridged Networks and Provider Backbone Bridged Networks. These networks provide additional mapping facilities to support hierarchies of VLANs allowing a provider a separate filtering database from customers. The conformance definitions of 5.3 have been extended to support S-VLANs. Clause 25 and Clause 26 describe how the mapping of S-VLANs into backbone service instances is accomplished within Provider Backbone Bridged Networks.

7.5 Ingress, forwarding, and egress rules

The relay function provided by each Bridge controls:

- a) Classification of each received frame as belonging to one and only one VLAN, and discard or acceptance of the frame for further processing on the basis of that classification and the received frame format, which can be one of three possible types:
 - 1) Untagged, and not explicitly identifying the frame as being associated with a particular VID;
 - 2) Priority-tagged, i.e., including a tag header conveying explicit priority information but not identifying the frames as being associated with a specific VID;
 - 3) VLAN-tagged, i.e., explicitly associating the frames with a particular VID.
 This aspect of relay implements the *ingress* rules.
- b) Implementation of the decisions governing where each frame is to be forwarded as determined by the current extent of the VLAN topology (7.3), station location information (7.4), and the additional management controls specified in Clause 8. This aspect of relay implements the *forwarding* rules.
- c) Queueing of frames for transmission through the selected Bridge Ports, management of the queued frames, selection of frames for transmission, and determination of the appropriate frame format type, VLAN-tagged or untagged. This aspect of relay implements the *egress* rules.

The structuring of the relay functionality into the implementation of ingress, forwarding, and egress rules constitutes a generic approach to the provision of VLAN functionality. All VLAN-aware Bridges can correctly forward received frames that are already VLAN-tagged. These are classified as belonging to the VLAN identified by the VID in the tag header. All VLAN-aware Bridges can also classify untagged and priority-tagged frames received on any given port as belonging to a specified VLAN. In addition to this default Port-based ingress classification, this standard specifies an optional Port-and-Protocol-based classification.

The classification of untagged and priority-tagged frames, and the addition or removal of tag headers, is performed in support of the EISS (6.9).

Frames that carry control information to determine the active topology and current extent of each VLAN, i.e., spanning tree and MVRP PDUs, and frames from other link constrained protocols, such as EAPOL and LLDP, are not forwarded. Permanently configured static entries in the filtering database (8.2, 8.3, and 8.12) ensure that such frames are discarded by the Forwarding Process (8.6).

NOTE—MRPDUs destined for any MRP application are forwarded or filtered depending on whether the application concerned is supported by the bridge, as specified in 8.12.

The forwarding rules specified for VLAN-tagged frames facilitate the interoperation of bridges conformant to this standard with end stations that directly support attachment of MAC service users to VLANs by

transmitting VLAN-tagged frames, and with Bridges that are capable of additional proprietary ingress classification methods.

Frames transmitted on a given LAN by a VLAN-aware Bridge for a given VLAN shall be either

- d) All untagged; or
- e) All VLAN tagged with the same VID.

8. Principles of bridge operation

This clause

- a) Explains the principal elements of the operation of VLAN-aware Bridges, and the operation of VLAN-aware bridge components within systems, and lists the supporting functions.
- b) Establishes a Bridge architecture that governs the provision of these functions.
- c) Provides a model of Bridge operation in terms of processes and entities that support the functions.
- d) Details the addressing requirements in a Bridged Local Area Network.
- e) Specifies the addressing of Entities in a Bridge.

NOTE—The provisions of this clause subsume the provisions of Clause 7 of IEEE Std 802.1D-2004.

8.1 Bridge operation

The principal elements of Bridge operation are

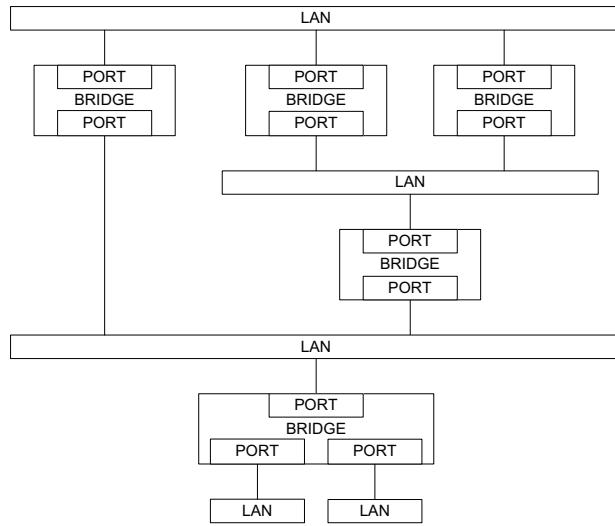
- a) Relay and filtering of frames (8.1.1).
- b) Maintenance of the information required to make frame filtering and relaying decisions (8.1.2).
- c) Management of the above (Clause 12).

8.1.1 Relay

A MAC Bridge relays individual MAC user data frames between the separate MACs of the bridged LANs connected to its Ports. The functions that support relaying of frames and maintain the Quality of Service are

- a) Frame reception (8.5).
- b) Discard on received frame in error (6.5.2).
- c) Discard of frames that do not carry user data (6.7).
- d) Priority and drop eligibility decoding from a VLAN TAG, if present, and regeneration of priority, if required (6.9).
- e) Classification of each received frame to a particular VLAN (6.9).
- f) Frame discard to support management control over the active topology of each VLAN (8.6.2).
- g) Frame discard to suppress loops in the physical topology of the network (8.6.1).
- h) Frame discard following the application of filtering information (8.6.3).
- i) Metering of frames, potentially marking as drop eligible or discarding frames exceeding bandwidth limits (8.6.5).
- j) Forwarding of received frames to other Bridge Ports (8.6.4).
- k) Selection of traffic class and queuing of frames by traffic class (8.6.6).
- l) Frame discard to ensure that a maximum bridge transit delay is not exceeded (6.5.6, 8.6.7).
- m) Preferential discard of drop eligible frames to preserve QoS for other frames (8.6.7).
- n) Selection of queued frames for transmission (8.6.8).
- o) Mapping of service data units and Frame Check Sequence recalculation, if required (6.5.7).
- p) Frame discard on transmittable service data unit size exceeded (6.5.8).
- q) Selection of outbound access priority (6.5.9).
- r) Frame transmission (8.5).

Figure 8-1 gives an example of the physical topology of a Bridged Local Area Network.

**Figure 8-1—A Bridged Local Area Network**

8.1.2 Filtering and relaying information

A Bridge maintains filtering and relaying information for the following purposes:

- a) Duplicate frame prevention: to maintain a loop-free active topology for each VLAN;
- b) Traffic segregation: to separate communication by different sets of network users;
- c) Traffic reduction: to confine frames to the path(s) between their source and destination(s);
- d) Traffic expediting: to classify frames in order to expedite time-critical traffic;
- e) Frame format conversion: to tag or untag as appropriate for the destination LAN and stations.

8.1.3 Duplicate frame prevention

A Bridge filters frames, i.e., does not relay frames received by a Bridge Port to other Ports on that Bridge, in order to prevent the duplication of frames (6.5.4). The functions that support the use and maintenance of information for this purpose are

- a) Configuration and calculation of one or more spanning tree active topologies.
- b) In MST Bridges, explicit configuration of the relationship between VIDs and spanning trees (8.9).

8.1.4 Traffic segregation

A Bridge can filter frames to confine them to LANs that belong to the VLAN to which they are assigned and, thus, define the VLAN's maximum extent (7.3). The functions that support the use and maintenance of information for this purpose are

- a) Configuration of a PVID for each Port, to associate a VID with untagged and priority-tagged received frames (6.9.1), and parameters for Protocol VLAN Classification (6.12) if implemented;
- b) Configuration of Static VLAN Registration Entries (8.8.2);
- c) Configuration of the Enable Ingress Filtering parameter to enable or disable application of Static VLAN Registration Entries to received frames.

A Bridge can filter frames to partially partition a Virtual Bridged Local Area Network. Frames assigned to any given VID and addressed to specific end stations or groups of end stations can be excluded from relay to certain Bridge Ports. The functions that support the use and maintenance of information for this purpose are

- d) Permanent configuration of Reserved Addresses (Table 8-1, Table 8-2, and Table 8-3);
- e) Configuration of Static Filtering Entries (8.8.1) and MAC Address Registration Entries (8.8.4).

NOTE—The use of VLANs is generally less error prone and is preferred to filtering using destination addresses if a Bridged Local Area Network is to be partitioned for reasons of scale, efficiency, management, or security. Destination address filtering is the only mechanism available to Bridges that are not VLAN-aware.

8.1.5 Traffic reduction

A Bridge can filter frames to confine them to LANs either that have end stations attached to their assigned VLAN or that connect those LANs and, thus, define the current practical extent of the VLAN (7.4). LANs not attaching to or forming part of the path between the source and the destination(s) of any given communication do not have to support the transmission of related frames, potentially improving the quality of the MAC service for other communications. The functions that support the use and maintenance of information for this purpose are

- a) Automatic learning of dynamic filtering information for unicast destination addresses through observation of source addresses of frames;
- b) Ageing out or flushing of dynamic filtering information that has been learned to support the movement of end stations and changes in active topology;
- c) Automatic inclusion and removal of Bridge Ports in the VLAN, through configuration of Dynamic VLAN Registration Entries by means of MVRP (8.8.5 and 11.2);
- d) Explicit configuration of management controls associated with the operation of MVRP by means of Static VLAN Registration Entries (8.8.2 and 11.2);
- e) Automatic configuration of MAC Address Registration Entries by means of MMRP exchanges;
- f) Explicit configuration of the management controls associated with the operation of MMRP by means of MAC Address Registration Entries.

8.1.6 Traffic expediting

A Bridge classifies frames into traffic classes in order to expedite transmission of frames generated by critical or time-sensitive services. The function that supports the use and maintenance of information for this purpose is

- a) Explicit configuration of traffic class information associated with the Ports of the Bridge.

8.1.7 Conversion of frame formats

A Bridge adds and removes tag headers (9.3) from frames and performs the associated frame translations that may be required. The function that supports the use and maintenance of information for this purpose is

- a) Explicit configuration of tagging requirements on transmission for each Port (8.8.2, 6.9.2).

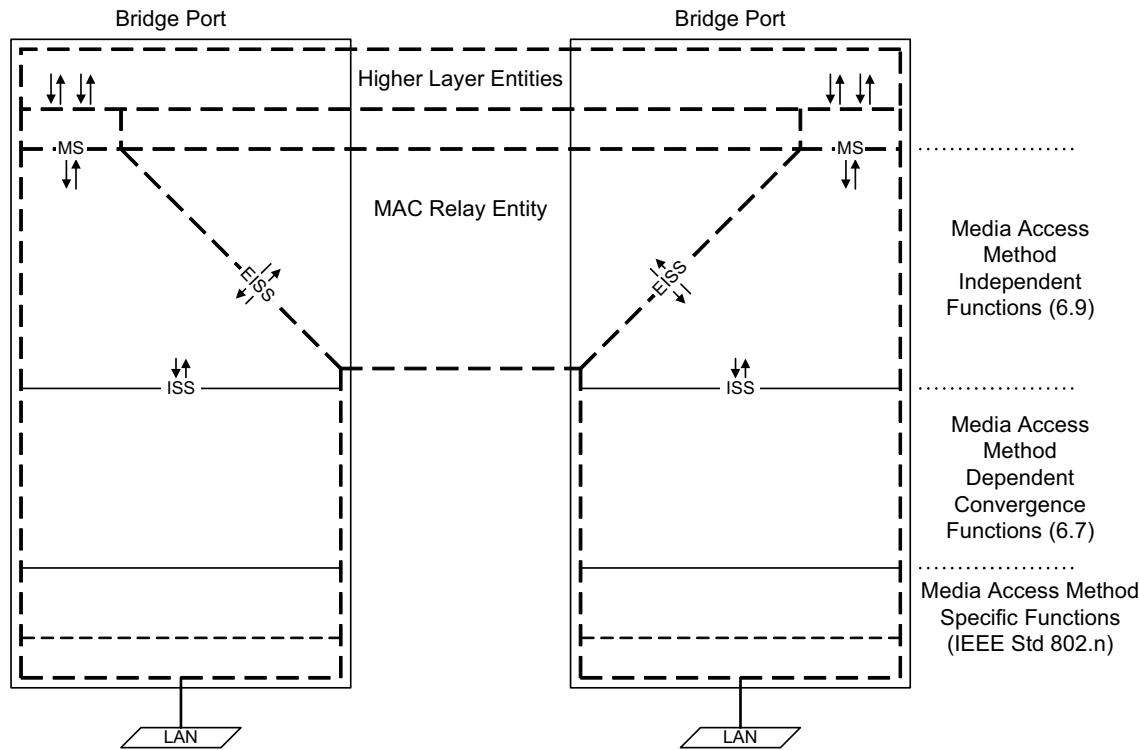
NOTE—As all incoming frames, including priority-tagged frames, are classified as belonging to a VLAN, the transmitting Port transmits VLAN-tagged frames or untagged frames. Hence, a station sending a priority-tagged frame via a Bridge will receive a response that is either VLAN-tagged or untagged, as described in 8.5.

8.2 Bridge architecture

A Bridge comprises at least one bridge component. A bridge component comprises

- a) A MAC Relay Entity that interconnects the Bridge's Ports;
- b) At least two Ports;
- c) Higher layer entities, including at least a Spanning Tree Protocol Entity.

The VLAN-aware Bridge architecture is illustrated in Figure 8-2. The MAC Relay Entity handles the media access method independent functions of relaying frames among Bridge Ports, filtering frames, and learning filtering information. It uses the Enhanced Internal Sublayer Service (EISS) (6.8, 6.9) provided by each Bridge Port.



NOTE—The notation “IEEE Std 802.n” in this figure indicates that the specifications for these functions can be found in the relevant standard for the media access method concerned; for example, n would be 3 (IEEE Std 802.3) in the case of Ethernet.

Figure 8-2—VLAN-aware Bridge architecture

Each Bridge Port also functions as an end station providing one or more instances of the MAC Service. Each instance of the MAC Service is provided to a distinct LLC Entity that supports protocol identification, multiplexing, and demultiplexing, for PDU transmission and reception by one or more higher layer entities.

NOTE 1—In most cases, each Port provides a single instance of the MAC Service, to an LLC Entity that supports all Higher Layer Entities that require a point of attachment to the Port. Further instances are only provided when the specifications of the Higher Layer Entities require the use of different instances of the MAC service or of different source addresses.

An LLC Entity for each Bridge Port shall use an instance of the MAC Service provided for that Port to support the operation of LLC Type 1 procedures in order to support the operation of the Spanning Tree

Protocol Entity. Bridge Ports may support other types of LLC procedures for use by other protocols, such as protocols providing Bridge Management (8.12).

NOTE 2—For simplicity of specification, this standard refers to a single LLC Entity that can provide both the procedures specified by ISO/IEC 8802-2 and EtherType protocol discrimination in the cases where the media access method for the attached LAN supports the latter.

If the Bridge Port can be directly attached to an IEEE 802 LAN, an instance of the MAC for that LAN type is permanently associated with the Port, handles the media access method specific functions (MAC protocol and procedures), and provides an instance of the Internal Sublayer Service (ISS) as specified in 6.6 to support frame transmission and reception by the other processes and entities that compose the Port.

Figure 8-2 illustrates a Bridge with two Ports, each directly connected to a LAN.

8.3 Model of operation

The model of operation is simply a basis for describing the functionality of the MAC Bridge. It is in no way intended to constrain real implementations of a MAC Bridge; these may adopt any internal model of operation compatible with the externally visible behavior that this standard specifies. Conformance of equipment to this standard is purely in respect of observable protocol.

The processes and entities that model the operation of a Bridge Port include

- a) A Bridge Port Transmit and Receive Process (8.5) that:
 - 1) Receives and transmits frames from and to the attached LAN (8.5, 6.6, 6.8, 6.9);
 - 2) Can filter received frames if the VLAN tag is absent, or present, or conveys a null VID;
 - 3) Classifies received frames into VLANs, assigning each a VID value;
 - 4) Determines the format, VLAN-tagged or untagged, of transmitted frames;
 - 5) Delivers and accepts frames to and from the MAC Relay Entity and LLC Entities;
- b) The LLC Entity or Entities that support Higher Layer Entities such as:
 - 1) Spanning Tree Protocol;
 - 2) Multiple Registration Protocol;
 - 3) Bridge Management;
 - 4) Linktrace response.
- c) Zero or more Connectivity Fault Management (CFM) Entities, each of which can be one of the following:
 - 1) A Maintenance association End Point (MEP);
 - 2) A Maintenance association Intermediate Point (MIP); or
 - 3) A Linktrace Output Multiplexer.

The processes and entities that model the operation of the MAC Relay Entity are

- d) The Forwarding Process (8.6), that:
 - 1) Enforces a loop-free active topology for frames for all VLANs (8.1.3, 8.4, 8.6.1);
 - 2) Filters frames using their VID and destination MAC Addresses (8.1.4, 8.6.2, 8.6.3, 8.6.4);
 - 3) Optionally, classifies and meters received frames that are to be relayed to other Bridge Ports (8.6.5);
 - 4) Forwards received frames that are to be relayed to other Bridge Ports (8.6.6, 8.6.7, 8.6.8);
- e) The Learning Process (8.7), that observes the source addresses of frames received on each Port, and updates the Filtering Database (8.1.5, 8.4);
- f) The Filtering Database (8.8), that holds filtering information and supports queries by the Forwarding Process as to whether frames with given values of VID and destination MAC Address field can be forwarded to a given Port.

Figure 8-3 illustrates a single instance of frame relay between the Ports of a Bridge with two Ports.

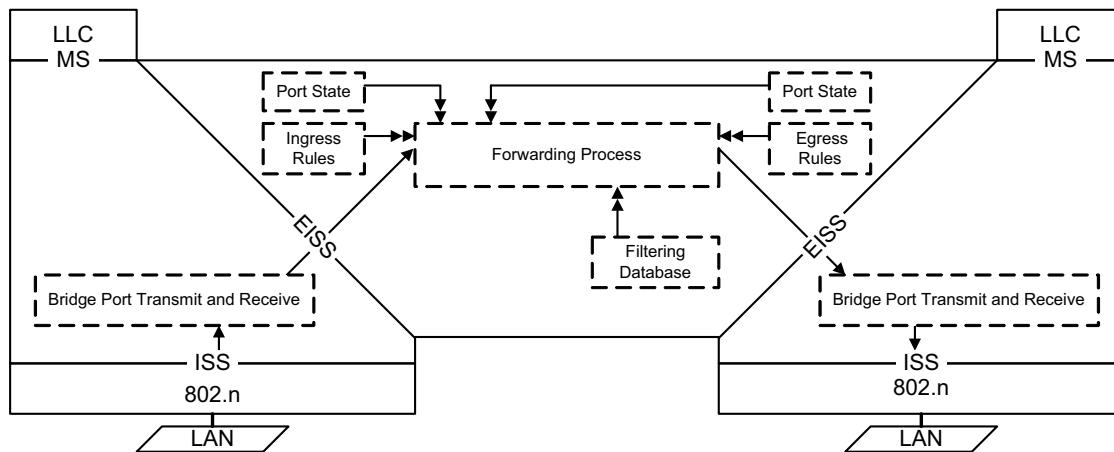


Figure 8-3—Relaying MAC frames

Figure 8-4 illustrates the inclusion of information carried by a single frame, received on one of the Ports of a Bridge with two Ports, in the Filtering Database.

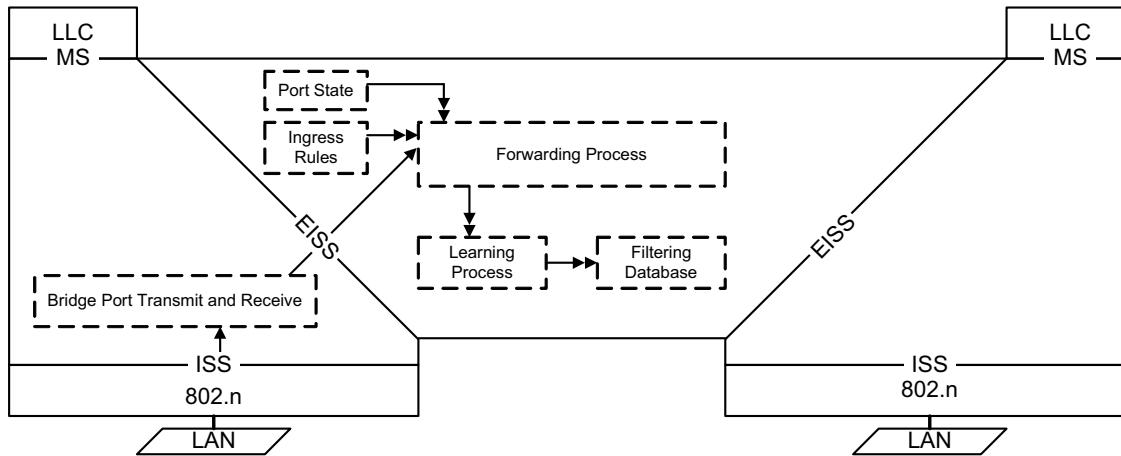


Figure 8-4—Observation of network traffic

Figure 8-5 illustrates the operation of the Spanning Tree Protocol Entity.

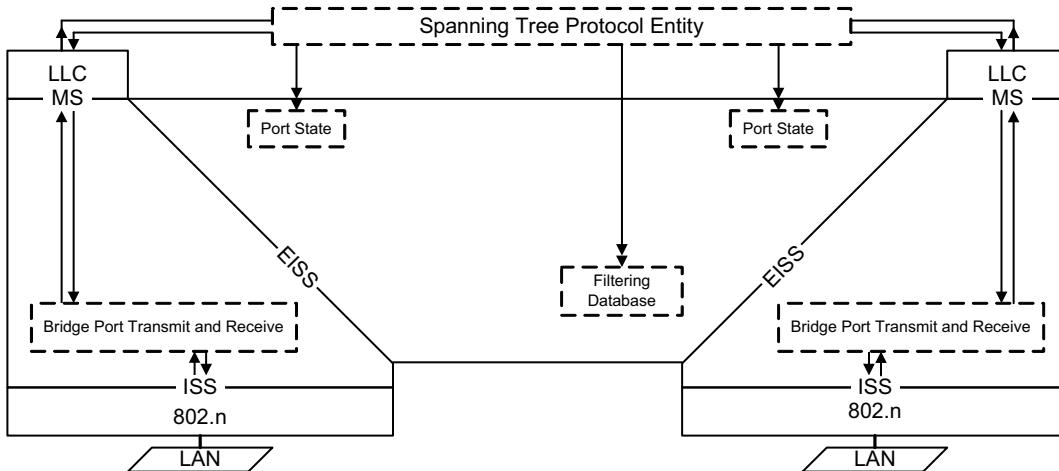
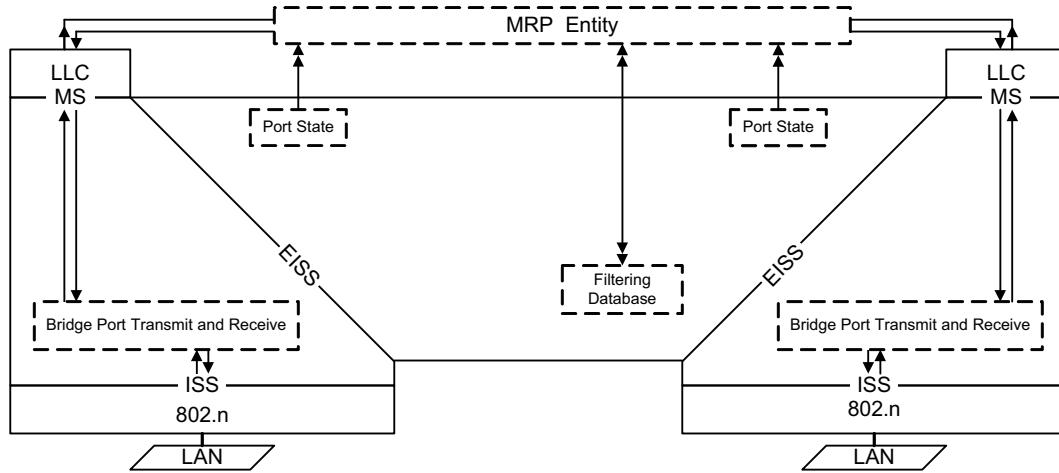
**Figure 8-5—Operation of Spanning Tree protocol**

Figure 8-6 illustrates the operation of the Multiple Registration Protocol (MRP) Entity (8.10).

**Figure 8-6—Operation of MRP**

Higher Layer Entities that require only one point of attachment for the Bridge as a whole may attach to an LLC Entity that uses an instance of the MAC Service provided by a Management Port. A Management Port does not use an instance of the ISS to attach to a network but uses the Bridge Port Transmit and Receive Process and the MAC Relay Entity to provide connectivity to other Bridge Ports and the attached LANs.

NOTE—Management port functionality may also be provided by an end station connected to an IEEE 802 LAN that is wholly contained within the system that incorporates the Bridge. The absence of external connectivity to the LAN ensures that access to the management port through the bridge's relay functionality can be assured at all times.

Figure 8-7 illustrates the reception and transmission by an Entity attached to a Management Port.

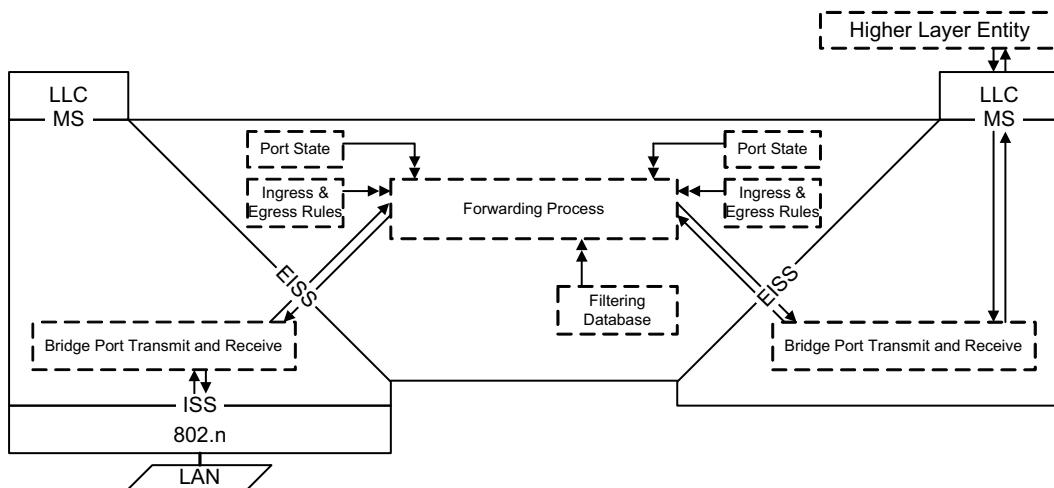


Figure 8-7—Management Port transmission and reception

8.4 Active topologies, learning, and forwarding

An *active topology* is the connectivity provided, for frames with a specified set of VID, destination address, and source address values, by interconnecting the LANs and bridges in a bridged network with *forwarding* Bridge Ports.

The distributed spanning tree algorithms and protocols, i.e. the Rapid Spanning Tree Protocol (RSTP, Clause 13), and the Multiple Spanning Tree Protocol (MSTP, Clause 13), construct one or more active topologies, each simply and fully connected for frames being associated with any given VID. The *forwarding* and *learning* performed by each Bridge Port for each spanning tree is dynamically managed by RSTP or MSTP to prevent temporary loops and reduce excessive traffic in the network while minimizing denial of service following any change in the *physical topology* of the network.

Provider Backbone Bridge Traffic Engineering enables construction of active topologies by the external agent that is responsible for setting up Ethernet Switched Paths (ESPs).

Any port that is not enabled, i.e., has MAC_Operational (6.6.2) False or has been excluded from the active topology by management setting of the Administrative Bridge Port State to Disabled, has both forwarding and learning disabled for all spanning trees and ESPs.

If the Bridge Port is enabled, PBB-TE disables learning and enables forwarding for all frames allocated to each ESP-VID. An external agent manages the Filtering Database to control the forwarding of frames with particular values of ESP-VID and destination MAC address.

The term Port State summarizes per tree combinations of forwarding and learning, and any additional per tree variables used by a given spanning tree protocol to enforce the active topologies it has calculated, and is used by RSTP and MSTP as follows. Any port that has learning and forwarding disabled is assigned the Port State Discarding. A Port that has learning enabled but forwarding disabled has the Port State Learning, and a Port that both learns and forwards frames has the Port State Forwarding. However the RSTP and MSTP state machines (Clause 13) do not control the Port State directly but use independent forwarding and learning variables for each tree.

RSTP constructs a single spanning tree, the Common Spanning Tree (CST), and maintains a single Port State for each Bridge Port. SST Bridges allocate all frames to that single spanning tree irrespective of their VLAN classification or source and destination MAC addresses.

MSTP constructs multiple spanning trees, the Common and Internal Spanning Tree (CIST) and additional Multiple Spanning Tree Instances (MSTIs), and maintains a Port State for each spanning tree for each Port. An MST Bridge allocates all frames classified as belonging to a given VLAN to the CIST or to one of the MSTIs using the MST Configuration Table. An MSTID of 0 identifies the CIST. A reserved MSTID value (TE-MSTID) is used to identify VIDs for use by PBB-TE with ESPs. A single VID is used to identify frames assigned to any given VLAN that is supported by the CIST or an MSTI. That VID is used by the Forwarding Process (8.6) to identify the spanning tree for the relayed frame, and thus identifies the applicable Port State.

Figure 8-5 illustrates the operation of the Spanning Tree Protocol Entity, which operates the Spanning Tree algorithm and its related protocols, and its modification of Port state information as part of determining the active topology of the network.

Figure 8-3 illustrates the Forwarding Process's use of the Port State: first, for a Port receiving a frame, to determine whether the received frame is to be relayed through any other Ports; and second, for another Port in order to determine whether the relayed frame is to be forwarded through that particular Port.

Figure 8-4 illustrates the use of the Port state information for a Port receiving a frame, in order to determine whether the station location information is to be incorporated in the Filtering Database.

8.5 Bridge Port Transmit and Receive

The Bridge Port Transmit and Receive process supports the attachment of the Bridge Port to a network. It comprises the following functions:

- a) In a VLAN-aware Bridge component, mapping between the EISS (6.8) provided to the MAC Relay Entity, and the ISS (6.6), adding, recognizing, interpreting, and removing VLAN tags as specified in 6.9 and illustrated in Figure 8-8;
- b) In a VLAN-unaware Bridge component, an optional shim (6.1.5) that uses and provides the ISS (6.6), recognizing and interpreting VLAN tags as specified in 6.20 and illustrated in Figure 8-9.
- c) Connectivity, as specified in 8.5.1 (for non-TPMR Bridges) or 8.5.2 (for TPMRs), between the following ISS access points:
 - 1) That provided by the MAC Entity for the LAN attached to the Port, as specified in 6.7;
 - 2) That supporting the MAC Relay Entity; and
 - 3) One or more that support Higher Layer Entities attached to the Port (8.5.3).

A single Port for a Bridge, known as the Management Port, may support Higher Layer Entities without providing a point of attachment to a LAN (see Figure 8-7).

8.5.1 Bridge Port connectivity

Each M_UNITDATA.indication provided by the ISS access point for an attached LAN [(1) in Figure 8-8] shall result in a corresponding M_UNITDATA.indication with identical parameters at each of the access points supporting the MAC Relay and Higher Layer Entities [(2), (3a) and (3b)]. Each M_UNITDATA.request from the ISS access point supporting the MAC Relay Entity shall result in a corresponding M_UNITDATA.indication with identical parameters at each of the access points for the Higher Layer Entities [(3a) and (3b)], and a corresponding M_UNITDATA.request with identical parameters at the access point for the LAN [(1)]. Each M_UNITDATA.request from an ISS access point supporting a Higher Layer Entity shall result in a corresponding M_UNITDATA.indication with identical parameters at

the access points for the MAC Relay Entity, and at other access points for Higher Layer Entities, and a corresponding M_UNITDATA.request with identical parameters at the access point for the LAN.

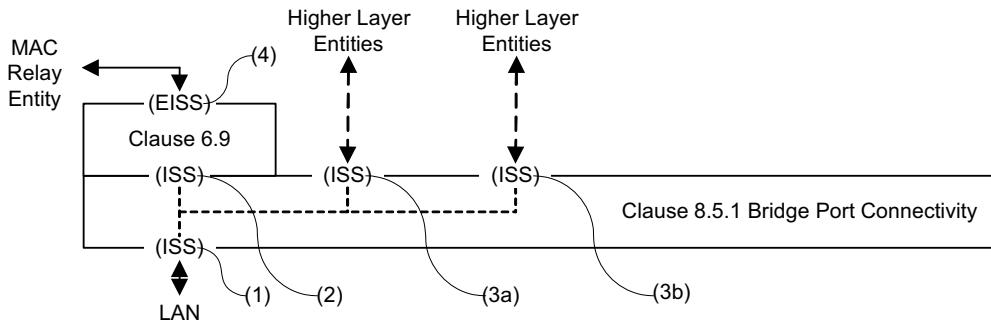


Figure 8-8—Bridge Port Transmit and Receive

NOTE—Figure 8-8 shows the relative position of the Bridge Port Transmit and Receive process and the VLAN tagging function (6.9) for a Customer Bridge or Provider Bridge that does not support CFM. For the relative positioning of these functions in a bridge supporting CFM, see Figure 22-4. For the relative positioning of these functions in a Backbone Edge Bridge, see Figure 26-2.

The MAC_Enabled, MAC_Operational, and operPointToPointMAC status parameters for the ISS access point for the MAC Relay Entity and Higher Layer Entities [(2), (3a), and (3b) in Figure 8-8] shall take the same value as that for the LAN (1) if that is present, and shall be True otherwise (i.e., if the Port is a Management Port).

The consequence of the above connectivity is that frames relayed to a Bridge Port are both submitted to that Port's MAC Service users and transmitted on the attached LAN (see 8.13.9).

8.5.2 TPMR Port connectivity

Each M_UNITDATA.indication provided by the ISS access point for an attached LAN [(1) in Figure 8-9] shall result in a corresponding M_UNITDATA.indication with identical parameters at each of the access points supporting the MAC Relay and Higher Layer Entities[(2), (3a), and (3b)].

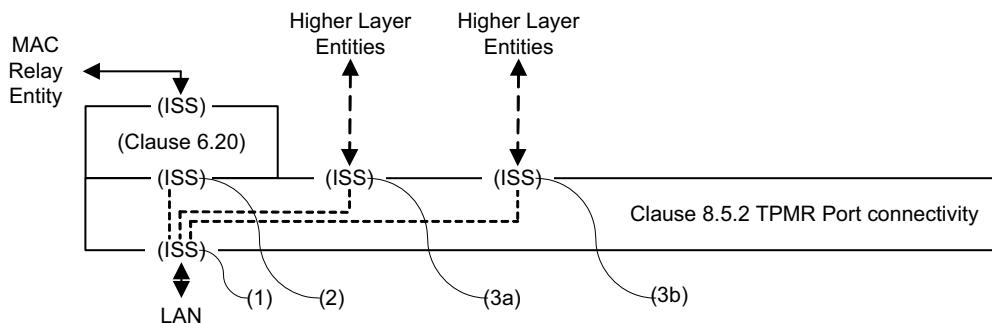


Figure 8-9—TPMR Port Transmit and Receive

Each M_UNITDATA.request from the ISS access point supporting the MAC Relay Entity (2) or a Higher Layer Entity [(3a) or (3b)] shall result in a corresponding M_UNITDATA.request with identical parameters at the access point for the LAN (1).

The MAC_Enabled, MAC_Operational, and operPointToPointMAC status parameters for the ISS access point for the MAC Relay Entity and Higher Layer Entities [(2), (3a), and (3b) in Figure 8-9] shall take the same value as for the LAN (1).

NOTE—These connectivity rules allow a higher layer entity to “listen” on both TPMR Ports to be able to determine the LAN on which a particular frame was received, and ensure that any frames sent by a higher layer entity are propagated only on the LAN associated with the Port on which the frame is transmitted. This differs from the propagation rules for other bridges, where frames transmitted by a higher layer entity are also submitted to the forwarding process, and frames transmitted by the Forwarding Process on a given Port are received by any higher layer entity associated with that Port.

8.5.3 Support of Higher Layer Entities

The MAC Service is provided to a Higher Layer Entity using one of ISS access points provided for that purpose by the Bridge Port Connectivity function (8.5.1) or the TPMR Port Connectivity function (8.5.2).

Each ISS M_UNITDATA.indication with a destination MAC address that is either the individual address of a MAC service access point (MSAP) provided by the Bridge Port or a group address used by the attached LLC Entity shall cause an MA_UNITDATA.indication at that MSAP with destination address, source address, MSDU, and priority parameters identical to those in the M_UNITDATA.indication. No other indications or frames give rise to indications to the MAC Service user.

Each MA_UNITDATA.request at the MSAP shall result in an M_UNITDATA.request at the ISS access point with identical destination address, source address, MSDU, and priority parameters. Each MA_UNITDATA.request at the MSAP with a destination MAC address that is either the individual address of the MSAP or a group address used by the attached LLC Entity shall also result in an MA_UNITDATA.indication at the MSAP with identical destination address, source address, MSDU, and priority parameters.

NOTE 1—The requirement to reflect self-addressed MA_UNITDATA.request primitives back as MA_UNITDATA.indications to the MSAP at which they were received exists because the MAC Service expects this behavior (ISO/IEC 15802-1) but the ISS does not expect it or provide it.

NOTE 2—Appropriate selection of the PVID for a Management Port facilitates attachment of an IP stack supporting an SNMP Management Agent to any selected VID relayed by the Bridge. This function is not supported in bridges that do not support the EISS, such as TPMRs.

NOTE 3—Higher Layer Entities attached to a Bridge Port can use the VLAN tag of received frames to achieve VLAN-awareness.

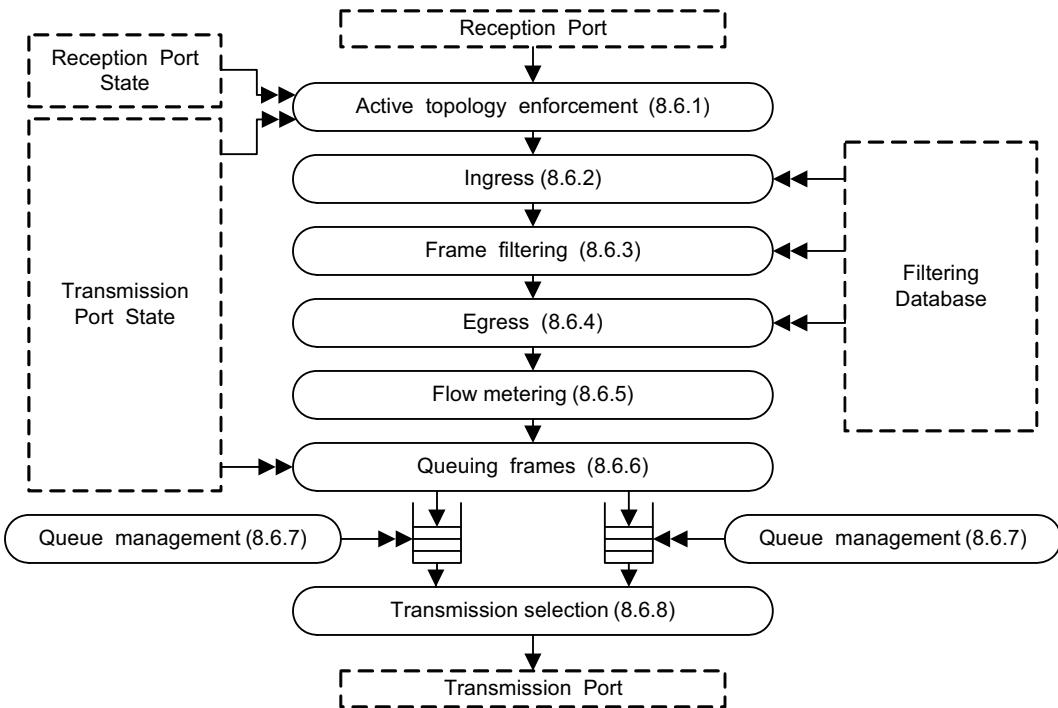
8.6 The Forwarding Process

Each frame submitted to the MAC Relay Entity shall be forwarded subject to the constituent functions of the Forwarding Process (Figure 8-10). Each function is described in terms of the action taken for a given frame received on a given Port (termed “the reception Port”). The frame can be forwarded for transmission on some Ports (termed “transmission Ports”) and discarded without being transmitted at the other Ports.

NOTE 1—IEEE Std 802.1Q, 2003 Edition, included frame formatting and FCS recalculation functions within the Forwarding Process. This standard now places those functions below the EISS interface, to allow the specification of additional methods for Bridge Port support of the EISS.

NOTE 2—The Forwarding Process models the MAC relay function and does not take into consideration what may occur once frames are passed to the Bridge Port for transmission. Conformant implementations of some media access methods can vary the transmission order in apparent violation of the transmission selection rules when observing frames on the medium. It may therefore not be possible to test conformance to this standard for some implementations simply by relating observed LAN traffic to the functionality of the forwarding process; tests also have to allow for the (conformant) behavior of the MAC.

Figure 8-3 illustrates the operation of the Forwarding Process in a single instance of frame relay between the Ports of a Bridge with two Ports.

**Figure 8-10—Forwarding Process functions**

8.6.1 Active topology enforcement

To prevent data loops and unwanted learning of source MAC addresses, the Forwarding Process determines the values (TRUE or FALSE) of the learning and forwarding controls (8.4) appropriate to each received frame and Bridge Port. If learning is true for the reception Port and ingress filtering (8.6.2) would not cause the received frame to be discarded, the source address and VID are submitted to the Learning Process. If forwarding is true for the reception Port, each Bridge Port, other than the reception Port, with forwarding true is identified as a potential transmission Port.

An SST Bridge supports a single active topology, the Common Spanning Tree (CST). For each Bridge Port, RSTP determines a single value for forwarding and a single value for learning (8.4) for all frames.

Bridges with MST or PBB-TE capabilities use the VID of the received frame to determine forwarding and learning for each Bridge Port for that frame as follows:

If the bridge supports PBB-TE, and the VID is an ESP-VID, forwarding is true and learning is false. Control over the active topology of each ESP is provided by configuration of static filtering entries in the Filtering Database (8.8.1, 25.10.1).

If the bridge uses MSTP to configure the CIST, and the VID identifies a VLAN assigned to the CIST, forwarding and learning are as determined by MSTP (Clause 13) for the CIST.

If the bridge uses MSTP to configure MSTIs, and the VID identifies a VLAN assigned to an MSTI calculated by MSTP, forwarding and learning are as determined by MSTP for that MSTI.

All bridges other than SST Bridges implement the MST Configuration Table (8.9.1). The use of VIDs to determine learning and forwarding, as required by this clause shall be consistent with that table as follows. All VIDs allocated by the MST Configuration Table to the CIST-MSTID (0x000) are assigned to the CIST,

all VIDs allocated to the TE-MSTID (0xFFE) are ESP-VIDs, and all VIDs allocated to other values of MSTID are assigned to the MSTI identified by that MSTID.

8.6.2 Ingress filtering

Each Port may support an Enable Ingress Filtering parameter. A frame received on a Port that is not in the member set (8.8.10) associated with the frame's VID shall be discarded if this parameter is set. The default value for this parameter is reset, i.e., Disable Ingress Filtering, for all Ports. Any Port that supports setting this parameter shall also support resetting it. The parameter may be configured by the management operations defined in Clause 12.

8.6.3 Frame filtering

The Forwarding Process takes filtering decisions, i.e., reduces the set of potential transmission Ports (8.6.1), for each received frame on the basis of

- a) Destination MAC Address;
- b) VID;
- c) The information contained in the Filtering Database for that MAC Address and VID;
- d) The default Group filtering behavior for the potential transmission Port (8.8.6);

in accordance with the definition of the Filtering Database entry types (8.8.1, 8.8.3, and 8.8.4). The required behavior is summarized in 8.8.6, 8.8.9, Table 8-7, Table 8-8, and Table 8-9.

Each of the Reserved MAC Addresses specified in Table 8-1 shall be permanently configured in the Filtering Database in C-VLAN components. Each of the Reserved MAC Addresses specified in Table 8-2 shall be permanently configured in the Filtering Database in S-VLAN components. Each of the Reserved MAC Addresses specified in Table 8-3 shall be permanently configured in the Filtering Database in TPMR components. The Filtering Database Entries for Reserved MAC Addresses shall specify filtering for all Bridge Ports and all VIDs. Management shall not provide the capability to modify or remove entries for Reserved MAC Addresses.

The addresses in Table 8-1, Table 8-2, and Table 8-3 determine the scope of propagation of PDUs within a bridged LAN, as follows:

- e) The **Nearest Bridge** group address is an address that no conformant TPMR component, S-VLAN component, C-VLAN component or IEEE 802.1D Bridge can forward. PDUs transmitted using this destination address, or any of the other addresses that appear in all three tables, can therefore travel no further than those stations that can be reached via a single individual LAN from the originating station. Hence the Nearest Bridge group address is also known as the Individual LAN Scope group address.
- f) The **Nearest non-TPMR Bridge** group address is an address that no conformant S-VLAN component, C-VLAN component or IEEE 802.1D Bridge can forward; however this address is relayed by a TPMR component. PDUs using this destination address, or any of the other addresses that appear in both Table 8-1 and Table 8-2 but not in Table 8-3, will be relayed by any TPMRs but will propagate no further than the nearest S-VLAN component, C-VLAN component or IEEE 802.1D Bridge.
- g) The **Nearest Customer Bridge** group address is an address that no conformant C-VLAN component or IEEE 802.1D Bridge can forward; however it is relayed by TPMR components and S-VLAN components. PDUs using this destination address, or any of the other addresses that appear in Table 8-1 but not in either Table 8-2 or Table 8-3, will be relayed by TPMR components and S-VLAN components but will propagate no further than the nearest C-VLAN component or IEEE 802.1D Bridge.

NOTE 1—The Reserved MAC Addresses specified in Table 8-1, Table 8-2, and Table 8-3 determine the span of connectivity for the protocol frames using the address; they do not identify the protocol. The protocols are identified by the EtherType field following the address fields.

NOTE 2—An instance of IEEE 802.1AX LACP intended to operate between non-TPMR Bridges, even through a TPMR, can use the Nearest non-TPMR Bridge group address. See K.2 for discussion of the use of TPMRs in a link using LACP.

Table 8-1—C-VLAN component Reserved addresses

Assignment	Value
Bridge Group Address, Nearest Customer Bridge group address ^a	01-80-C2-00-00-00
IEEE MAC-specific Control Protocols group address	01-80-C2-00-00-01
IEEE 802.3 Slow_Protocols_Multicast address	01-80-C2-00-00-02
Nearest non-TPMR Bridge group address	01-80-C2-00-00-03
IEEE MAC-specific Control Protocols group address	01-80-C2-00-00-04
Reserved for future standardization	01-80-C2-00-00-05
Reserved for future standardization	01-80-C2-00-00-06
Metro Ethernet Forum ELMI protocol group address ^b	01-80-C2-00-00-07
Provider Bridge Group Address	01-80-C2-00-00-08
Reserved for future standardization	01-80-C2-00-00-09
Reserved for future standardization	01-80-C2-00-00-0A
Reserved for future standardization	01-80-C2-00-00-0B
Reserved for future standardization	01-80-C2-00-00-0C
Provider Bridge MVRP Address	01-80-C2-00-00-0D
Individual LAN Scope group address ^c , Nearest Bridge group address	01-80-C2-00-00-0E
Reserved for future standardization	01-80-C2-00-00-0F

^aAs stated in 13.39, in a Provider Edge Bridge, a C-VLAN component supporting a single service instance (i.e. with a single Provider Edge Port) may forward (not filter) frames with the Nearest Customer Bridge Address as the destination address.

^bThis address is not exclusively reserved for this purpose; other uses are reserved for future standardization.

^cIt is intended that no IEEE 802.1 relay device will be defined that will forward frames that carry this destination address.

Table 8-2—S-VLAN component Reserved addresses

Assignment	Value
IEEE MAC-specific Control Protocols group address	01-80-C2-00-00-01
IEEE 802.3 Slow_Protocols_Multicast address	01-80-C2-00-00-02
Nearest non-TPMR Bridge group address	01-80-C2-00-00-03
IEEE MAC-specific Control Protocols group address	01-80-C2-00-00-04
Reserved for future standardization	01-80-C2-00-00-05
Reserved for future standardization	01-80-C2-00-00-06
Metro Ethernet Forum ELMI protocol group address ^a	01-80-C2-00-00-07
Provider Bridge Group Address	01-80-C2-00-00-08
Reserved for future standardization	01-80-C2-00-00-09
Reserved for future standardization	01-80-C2-00-00-0A
Individual LAN Scope group address ^b , Nearest Bridge group address	01-80-C2-00-00-0E

^aThis address is not exclusively reserved for this purpose; other uses are reserved for future standardization.

^bIt is intended that no IEEE 802.1 relay device will be defined that will forward frames that carry this destination address.

Table 8-3—TPMR component Reserved addresses

Assignment	Value
IEEE MAC-specific Control Protocols group address	01-80-C2-00-00-01
IEEE 802.3 Slow_Protocols_Multicast address	01-80-C2-00-00-02
IEEE MAC-specific Control Protocols group address	01-80-C2-00-00-04
Individual LAN Scope group address ^a , Nearest Bridge group address	01-80-C2-00-00-0E

^aIt is intended that no IEEE 802.1 relay device will be defined that will forward frames that carry this destination address.

8.6.4 Egress

The Forwarding Process shall queue each received frame to each of the potential transmission Ports (8.6.1, 8.6.3) that is present in the member set (8.8.10) for the frame's VID.

NOTE—The Forwarding Process is modeled as receiving a frame as the parameters of a data indication and transmitting through supplying the parameters of a data request. Queueing a frame awaiting transmission amounts to placing the parameters of a data request on an outbound queue.

8.6.5 Flow classification and metering

The Forwarding Process may apply flow classification and metering to frames received on a Bridge Port.

Flow classification identifies a subset of traffic (frames) that may be subject to the same treatment in terms of metering and forwarding. Flow classification rules may be based on:

- a) Destination MAC Address
- b) Source MAC address
- c) VID
- d) Priority

Frames classified using the same set of classification rules are subject to the same flow meter. The flow meter can change the drop_eligible parameter associated with each frame and can discard frames on the basis of the following parameters for each received frame and previously received frames, and the time elapsed since those frames were received:

- d) The received value of the drop_eligible parameter
- e) The mac_service_data_unit size

The flow meter shall not base its decision on the parameters of frames received on other Bridge Ports, or on any other parameters of those Ports. The metering algorithm described in the Metro Ethernet Forum Technical Specification MEF 10.2 should be used.

NOTE 1—Changing the value of the drop_eligible parameter may change the contents of the frame, depending on how the frame is tagged when transmitted, which may then require updating the frame_check_sequence. Mechanisms for conveying information from ingress to egress that the frame_check_sequence may require updating are implementation dependent.

NOTE 2—The flow meter described here can encompass a number of meters, each with a simpler specification. However, given the breadth of implementation choice permitted, further structuring to specify, for example, that frames can bypass a meter or are subject only to one of a number of meters provides no additional information.

NOTE 3—Although flow metering is applied after egress (Figure 8-10), the meter(s) operate per reception Port (see first sentence of 8.6.5), not per potential transmission Port(s).

8.6.6 Queuing frames

The Forwarding Process provides storage for queued frames, awaiting an opportunity to submit these for transmission. The order of frames received on the same Bridge Port shall be preserved for

- a) unicast frames with a given VID, priority, and destination address and source address combination;
- b) multicast frames with a given VID, priority, and destination address.

The Forwarding Process provides one or more queues for a given Bridge Port, each corresponding to a distinct traffic class. Each frame is mapped to a traffic class using the Traffic Class Table for the Port and the frame's priority. Traffic class tables may be managed. Table 8-4 shows the recommended mapping for the number of classes implemented, in implementations that do not support the credit-based shaper transmission selection algorithm (8.6.8.2). The requirements for priority to traffic class mappings in implementations that support the credit-based shaper transmission selection algorithm are defined in 34.5. Up to eight traffic classes may be supported, allowing separate queues for each priority.

NOTE—Different numbers of traffic classes may be implemented for different Ports. Ports with media access methods that support a single transmission priority, such as CSMA/CD, can support more than one traffic class.

In a congestion aware Bridge (Clause 30), the act of queuing a frame for transmission on a Bridge Port can result in the Forwarding Process generating a Congestion Notification Message (CNM). The CNM is injected back into the Forwarding Process (Active topology enforcement, 8.6.1) as if it had been received on that Bridge Port.

Table 8-4—Recommended priority to traffic class mappings

		Number of available traffic classes							
		1	2	3	4	5	6	7	8
Priority	0 (Default)	0	0	0	0	0	1	1	1
	1	0	0	0	0	0	0	0	0
	2	0	0	0	1	1	2	2	2
	3	0	0	0	1	1	2	3	3
	4	0	1	1	2	2	3	4	4
	5	0	1	1	2	2	3	4	5
	6	0	1	2	3	3	4	5	6
	7	0	1	2	3	4	5	6	7

NOTE—The rationale for these mappings is discussed in Annex I.

8.6.7 Queue management

A frame queued for transmission on a Port shall be removed from that queue

- a) Following a transmit data request. No further attempt shall be made to transmit the frame on that Port even if the transmission is known to have failed.
- b) If that is necessary to ensure that the maximum bridge transit delay (6.5.6) will not be exceeded at the time at which the frame would subsequently be transmitted.
- c) If the transmission Port is no longer forwarding for that frame (8.4, 8.6.1). 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).

The frame can be removed from the queue, and not subsequently transmitted

- d) If the time for which buffering is guaranteed has been exceeded for that frame
- e) By a queue management algorithm that attempts to improve the QoS provided by deterministically or probabilistically managing the queue depth based on the current and past queue depths.

Removal of a frame from a queue for any particular Port does not affect queuing of that frame for transmission on any other Port.

NOTE—Applicable queue management algorithms include RED (random early detection), and WRED (weighted random early detection) (IETF RFC 2309 (1998) [B20]). If the Bridge supports drop precedence, i.e., is capable of decoding or encoding the drop_eligible parameter from or to the PCP field of VLAN tags (6.9.3), the algorithm should exhibit a higher probability of dropping frames with drop_eligible True.

The probability of removing a frame with drop_eligible set shall not be less than that of removing a frame with drop_eligible False, all other conditions being equal. If a queue management algorithm is implemented, it should preferentially discard frames with drop_eligible True. If a Bridge supports encoding or decoding of

drop_eligible from the PCP field of a VLAN tag (6.9.3) on any of its Ports, then it shall implement a boolean parameter Require Drop Encoding on each of its Ports with default value FALSE. If Require Drop Encoding is True and the Bridge Port cannot encode particular priorities with drop_eligible, then frames queued with those priorities and drop_eligible True shall be discarded and not transmitted.

8.6.8 Transmission selection

For each Port, frames are selected for transmission on the basis of the traffic classes that the Port supports and the operation of the transmission selection algorithms supported by the corresponding queues on that Port. For a given Port and supported value of traffic class, frames are selected from the corresponding queue for transmission if and only if

- a) The operation of the transmission selection algorithm supported by that queue determines that there is a frame available for transmission; and
- b) For each queue corresponding to a numerically higher value of traffic class supported by the Port, the operation of the transmission selection algorithm supported by that queue determines that there is no frame available for transmission.

The strict priority transmission selection algorithm defined in 8.6.8.1 shall be supported by all Bridges as the default algorithm for selecting frames for transmission. The credit-based shaper transmission selection algorithm defined in 8.6.8.2 may be supported in addition to the strict priority algorithm. Further transmission selection algorithms, selectable by management means, may be supported as an implementation option so long as the requirements of 8.6.6 are met.

The Transmission Selection Algorithm Table for a given Port assigns, for each traffic class that the Port supports, the transmission selection algorithm that is to be used to select frames for transmission from the corresponding queue. Transmission Selection Algorithm Tables may be managed, and allow the identification of vendor-specific transmission selection algorithms. The transmission selection algorithms are identified in the Transmission Selection Algorithm Table by means of integer identifiers, as defined in Table 8-5.

Table 8-5—Transmission selection algorithm identifiers

Transmission selection algorithm	Identifier
Strict priority (8.6.8.1)	0
Credit-based shaper (8.6.8.2)	1
Reserved for future standardization	2–255
Vendor-specific	A four-octet integer, where the 3 most significant octets hold an OUI value, and the least significant octet holds an integer value in the range 0–255 assigned by the owner of the OUI.

In implementations that do not support the credit-based shaper transmission selection algorithm, the recommended default configuration for the Transmission Selection Algorithm Tables is to assign the strict priority transmission selection algorithm to all supported traffic classes. In implementations that support the credit-based shaper transmission selection algorithm, the recommended default configuration for the Transmission Selection Algorithm Tables is defined in 34.5.

8.6.8.1 Strict priority algorithm

For a given queue that supports strict priority transmission selection, the algorithm determines that there is a frame available for transmission if the queue contains one or more frames. The order in which frames are selected for transmission from the queue shall maintain the ordering requirement specified in 8.6.6.

8.6.8.2 Credit-based shaper algorithm

For a given queue that supports credit-based shaper transmission selection, the algorithm determines that a frame is available for transmission if the following conditions are all true:

- a) The queue contains one or more frames.
- b) The *transmitAllowed* signal for the queue is TRUE.

The order in which frames are selected for transmission from the queue shall maintain the ordering requirement specified in 8.6.6.

The following external parameters are associated with each queue that supports the operation of the credit-based shaper algorithm:

- c) ***portTransmitRate***. The transmission rate, in bits per second, that the underlying MAC service that supports transmission through the Port provides. The value of this parameter is determined by the operation of the MAC.
- d) ***idleSlope***. The rate of change of *credit*, in bits per second, when the value of *credit* is increasing (i.e., while *transmit* is FALSE). The value of *idleSlope* can never exceed *portTransmitRate*. The value of *idleSlope* for a given queue is determined by the value of the ***operIdleSlope(N)*** parameter for that queue, as defined in 34.3.

The following internal parameters are associated with each queue that supports the credit-based shaper algorithm:

- e) ***transmit***. Takes the value TRUE for the duration of a frame transmission from the queue; FALSE when any frame transmission from the queue has completed.
- f) ***credit***. The transmission credit, in bits, that is currently available to the queue. If, at any time, there are no frames in the queue, and the *transmit* parameter is FALSE, and *credit* is positive, then *credit* is set to zero.
- g) ***sendSlope***. The rate of change of *credit*, in bits per second, when the value of *credit* is decreasing (i.e., while *transmit* is TRUE). The value of *sendSlope* is defined as follows:

$$\text{sendSlope} = (\text{idleSlope} - \text{portTransmitRate})$$

- h) ***transmitAllowed***. Takes the value TRUE when the *credit* parameter is zero or positive; FALSE when the *credit* parameter is negative.

NOTE 1—The value of *credit* is naturally constrained by the operating parameters for the algorithm, and also by the operation of any higher priority queues that use this algorithm. The lowest value that *credit* can reach depends on *sendSlope* and the largest frame that will be transmitted from the queue. This largest frame size is a consequence of the type of traffic that is being transmitted using the queue, rather than a limit imposed by the algorithm or by management. The highest value that can be accumulated in *credit* depends on *idleSlope* and the length of time that the algorithm may have to wait when the queue is not empty and there is other traffic being transmitted through the Port; for any given queue, that length of time is bounded, as discussed in Annex L, which also illustrates how the algorithm operates under varying conditions, and how its operation affects the maximum latency that can be experienced by SR traffic.

NOTE 2—In order for the credit-based shaper algorithm to operate as intended, it is necessary for all traffic classes that support the algorithm to be numerically higher than any traffic classes that support the strict priority algorithm defined in

8.6.8.1. The mapping of priorities onto traffic classes, and the choice of traffic classes that support particular transmission selection algorithms, is defined in 34.5.

8.7 The Learning Process

The Learning Process receives the source MAC Addresses and VIDs of received frames from the Forwarding Process, subject to active topology enforcement (8.6.1) and the application of ingress filtering (8.6.2). The Learning Process is not invoked (8.6.1) for frames whose VID is an ESP-VID.

When invoked, the Learning Process shall create or update a Dynamic Filtering Entry (8.8.3) that specifies the reception Port for the frame's source address and VID, if and only if the source address is an Individual Address, i.e., is not a Group Address, the resulting number of entries would not exceed the capacity of the Filtering Database, and the filtering utility criteria for the receiving Bridge Port are met, as specified below.

If the Filtering Database is already filled to capacity, but a new entry would otherwise be made, then an existing entry may be removed to make room for the new entry.

The purpose of filtering utility criteria is to reduce the capacity requirements of the Filtering Database and to reduce the time for which service can be denied (6.5.1) by retaining filtering information learned prior to a change in the physical topology of the network. Filtering utility criteria shall be applied to the learning and retention of information for each Filtering Identifier (FID) (8.8.8). Enhanced filtering utility criteria may be implemented for any Bridge Port as specified below (8.7.2); if implemented, both the default (8.7.1) and the enhanced criteria shall be selectable by management.

8.7.1 Default filtering utility criteria

The default for a VLAN-aware Bridge is that the member set (8.8.10) for the frame's VID includes at least one Port.

NOTE—If the member set for a given VID is empty, that VID is not currently active, and the Bridge will filter all frames destined for that VID, regardless of their destination address.

8.7.2 Enhanced filtering utility criteria

The enhanced criteria are satisfied if at least one VID that uses the FID includes the reception Port and at least one other Port with a Port State of Learning or Forwarding in its member set, and:

- a) The operPointToPointMAC parameter is false for the reception Port; or
- b) Ingress for the VID is permitted through a third Port.

NOTE—The third port can, but is not required to, be in the member set.

Figure 8-4 illustrates the operation of the Learning Process in the inclusion of station location information carried by a single frame, received on one of the Ports of a Bridge, in the Filtering Database.

8.8 The Filtering Database

The Filtering Database supports queries by the Forwarding Process to determine whether received frames, with given values of destination MAC Address and VID, are to be forwarded through a given potential transmission Port (8.6.1, 8.6.3, 8.6.4). The Filtering Database contains filtering information in the form of filtering entries that are either

- a) Static, and explicitly configured by management action; or

- b) Dynamic, and automatically entered into the Filtering Database by the normal operation of the bridge and the protocols it supports.

Two entry types are used to represent static filtering information. The Static Filtering Entry represents static information in the Filtering Database for individual and for group MAC Addresses. It allows administrative control of

- c) Forwarding of frames with particular destination addresses; and
- d) The inclusion in the Filtering Database of dynamic filtering information associated with Extended Filtering Services, and use of this information.

The Filtering Database shall contain entries of the Static Filtering Entry type.

The Static VLAN Registration Entry represents all static information in the Filtering Database for VLANs. It allows administrative control of

- e) Forwarding of frames with particular VIDs;
- f) The inclusion/removal of tag headers in forwarded frames; and
- g) The inclusion in the Filtering Database of dynamic VLAN membership information, and use of this information.

The Filtering Database may contain entries of the Static VLAN Registration Entry type.

Static filtering information is added to, modified, and removed from the Filtering Database only under explicit management control. It shall not be automatically removed by any ageing mechanism. Management of static filtering information may be carried out by use of the remote management capability provided by Bridge Management (8.12) using the operations specified in Clause 12.

Four entry types are used to represent dynamic filtering information:

- h) Each Dynamic Filtering Entry specifies the Port through which a frame with a given individual MAC Address and VID has been or can be received. Dynamic Filtering Entries can be created and updated by the Learning Process (8.7). Entries that are updated by the Learning Process are subject to ageing and removal by the Filtering Database.
- i) MAC Address Registration Entries support the registration of MAC Addresses. They are created, updated, and removed by MMRP in support of Extended Filtering Services (8.8.4, and Clause 10). If the Restricted_MAC_Address_Registration management control (10.12.2.3) is TRUE, then the creation of a MAC Address Registration Entry is not permitted unless a Static Filtering Entry exists that permits dynamic registration for the Group concerned.
- j) Dynamic VLAN Registration Entries are used to specify the Ports on which VLAN membership has been dynamically registered. They are created, updated, and removed by MVRP, in support of automatic VLAN membership configuration (Clause 11), subject to the state of the Restricted_VLAN_Registration management control (11.2.3.2.3). If the value of this control is TRUE, then the creation of a Dynamic VLAN Registration Entry is not permitted unless a Static VLAN Registration Entry exists that permits dynamic registration for the VID concerned.
- k) Dynamic Reservation Entries are used to specify the Ports on which stream reservations have been made. They are created, updated, and removed by the operation of SRP defined in Clause 35.

Static Filtering Entries and MAC Address Registration Entries comprise

- l) A MAC Address specification;
- m) A VLAN Identifier (VID);
- n) A Port Map, with a control element for each outbound Port to specify filtering for that MAC Address specification and VID.

Dynamic Filtering Entries comprise

- o) A MAC Address specification;
- p) A locally significant Filtering Identifier (FID; see 8.8.8);
- q) A Port Map, with a control element for each outbound Port to specify filtering for that MAC Address specification for the VID(s) allocated to that FID.

The Port Map in Dynamic Filtering Entries may contain a **connection_identifier** (8.8.12) for each outbound Port.

Static and Dynamic VLAN Registration Entries comprise

- r) A VLAN Identifier (VID);
- s) A Port Map, with a control element for each outbound Port to specify filtering for the VID.

Dynamic Reservation Entries comprise

- t) A MAC Address specification;
- u) A VID;
- v) A Port Map, with a control element for each outbound Port to specify filtering for that MAC Address specification in the VLAN specified by the VID.

Dynamic filtering information may be read by use of the remote management capability provided by Bridge Management (8.12) using the operations specified in Clause 12.

The Filtering Services supported by a Bridge (Basic and Extended Filtering Services) determine the default behavior of the Bridge with respect to the forwarding of frames destined for group MAC Addresses. In Bridges that support Extended Filtering Services, the default forwarding behavior for group MAC Addresses, for each Port, and for each VID, can be configured both statically and dynamically by means of Static Filtering Entries and/or MAC Address Registration Entries that can carry the following MAC Address specifications:

- w) All Group Addresses, for which no more specific Static Filtering Entry exists;
- x) All Unregistered Group Addresses (i.e., all group MAC Addresses for which no MAC Address Registration Entry exists), for which no more specific Static Filtering Entry exists.

NOTE 1—The All Group Addresses specification in item w), when used in a Static Filtering Entry with an appropriate control specification, provides the ability to configure a Bridge that supports Extended Filtering Services to behave as a Bridge that supports only Basic Filtering Services on some or all of its Ports. This might be done for the following reasons:

- The Ports concerned serve “legacy” devices that wish to receive multicast traffic, but are unable to register Group membership;
- The Ports concerned serve devices that need to receive all multicast traffic, such as routers or diagnostic devices.

The Filtering Database shall support the creation, updating, and removal of Dynamic Filtering Entries by the Learning Process (8.7). In Bridges that support Extended Filtering Services, the Filtering Database shall support the creation, updating, and removal of MAC Address Registration Entries by MMRP (Clause 10).

In Bridges that support SRP, the Filtering Database shall support the creation, updating, and removal of Dynamic Reservation Entries by SRP (Clause 35).

Figure 8-3 illustrates use of the Filtering Database by the Forwarding Process in a single instance of frame relay between the Ports of a Bridge with two Ports.

Figure 8-4 illustrates the creation or update of a dynamic entry in the Filtering Database by the Learning Process. The entries in the Filtering Database allow MAC Address information to be learned independently for each VID or set of VIDs, by relating a MAC Address to the VID or set of VIDs on which that address was learned.

NOTE 2—The ability to create VLAN-dependent Filtering Database entries allows a Bridge to support

- Multiple end stations with the same individual MAC Address residing on different VLANs;
- End stations with multiple interfaces, each using the same individual MAC Address, as long as not more than one end station or interface that uses a given MAC Address resides in a given VLAN.

Figure 8-5 illustrates the operation of the Spanning Tree Protocol Entity (8.10), which operates the Spanning Tree Algorithm and Protocol, and its notification of the Filtering Database of changes in active topology signaled by that protocol.

There are no standardized constraints on the size of the Filtering Database in an implementation for which conformance to this standard is claimed. The PICS Proforma in Annex A requires the following to be specified for a given implementation:

- y) The total number of entries (Static Filtering Entries, Dynamic Filtering Entries, MAC Address Registration Entries, Static VLAN Registration Entries, Dynamic VLAN Registration Entries, and Dynamic Reservation Entries) that the implementation of the Filtering Database can support, and
- z) Of that total number, the total number of VLAN Registration Entries (static and dynamic) that the Filtering Database can support.

8.8.1 Static Filtering Entries

A Static Filtering Entry specifies

- a) A MAC Address specification, comprising
 - 1) An Individual MAC Address; or
 - 2) A group MAC Address; or
 - 3) All Individual Addresses, for which no more specific Static Filtering Entry exists; or
 - 4) All Group Addresses, for which no more specific Static Filtering Entry exists; or
 - 5) All Unregistered Group Addresses, for which no more specific Static Filtering Entry exists.
- b) A VLAN identifier specification, comprising:
 - 1) The VID of a specific VLAN to which the static filtering information applies; or
 - 2) The wildcard VID (Table 9-2), indicating that the static filtering information applies to all VIDs for which no specific Static Filtering Entry exists.
- c) A Port Map, containing a control element for each outbound Port, specifying that a frame with a destination MAC Address and VID that meets this specification is to be
 - 1) Forwarded, independently of any dynamic filtering information held by the Filtering Database; or
 - 2) Filtered, independently of any dynamic filtering information; or
 - 3) Forwarded or filtered on the basis of dynamic filtering information, or on the basis of the default Group filtering behavior for the outbound Port (8.8.6) if no dynamic filtering information is present specifically for the MAC Address.

All Bridges shall have the capability to support the first two values for the MAC Address specification, both values of the VLAN identifier specification, and all three values for each control element for all Static Filtering Entries [i.e., shall have the capability to support item a1), item a2), item b1), item b2), item c1), item c2), and item c3)]. All Bridges supporting PBB-TE shall, in addition, support items a3) and a4) for

ESP-VIDs (8.9) and shall always have in their filtering databases Static Filtering Entries specifying items a3) and a4), and a Port Map specifying filtering for each outbound port, for every ESP-VID.

A Bridge that supports Extended Filtering Services shall have the capability to support items a1), a2), a4), and a5) for the MAC Address specification and all three control element values for all Static Filtering Entries.

For a given MAC Address specification, a separate Static Filtering Entry with a distinct Port Map may be created for each VID from which frames are received by the Forwarding Process.

In addition to controlling the forwarding of frames, Static Filtering Entries provide the Registrar Administrative Control values for MMRP (Clause 10). Static configuration of forwarding of a specific destination MAC address to an outbound port indicates Registration Fixed on that port, which indicates a desire to receive the specific frames even in the absence of dynamic information. Static configuration of filtering of frames that might otherwise be sent to an outbound port indicates Registration Forbidden. The absence of a Static Filtering Entry for a MAC address, or the configuration of forwarding or filtering on the basis of dynamic filtering information, indicates Normal Registration.

8.8.2 Static VLAN Registration Entries

A Static VLAN Registration Entry specifies

- a) The VID to which the static filtering information applies;
- b) A Port Map, consisting of a control element for each outbound Port, specifying
 - 1) The Registrar Administrative Control values for MVRP (Clause 11) for the VID. In addition to providing control over the operation of MVRP, these values can also directly affect the forwarding behavior of the Bridge, as described in 8.8.10. The values that can be represented are
 - i) Registration Fixed; or
 - ii) Registration Forbidden; or
 - iii) Normal Registration.
 - 2) Whether frames are to be VLAN-tagged or untagged when transmitted. The entries in the Port Map that specify untagged transmission compose the untagged set for the VID. The untagged set is empty if no Static VLAN Registration Entry exists for the VID.

A separate Static VLAN Registration Entry with a distinct Port Map may be created for each VID. All Bridges shall be capable of supporting all values for each control element for all Static VLAN Registration Entries; however, the ability to support more than one untagged VID on egress on any given Port is optional (see 5.4).

NOTE 1—In other words, it is possible to configure any VID as untagged on egress, but it is not necessarily possible to configure more than one VID as untagged on egress. It is an implementation option as to whether only a single untagged VID per Port on egress is supported, or whether multiple untagged VIDs per Port on egress are supported. When a single untagged VID per Port is supported, that VID is not necessarily the same as the PVID.

The initial state of the Permanent Database contains a Static VLAN Registration Entry for the VID corresponding to the Default PVID (Table 9-2). The Port Map in this entry specifies Registration Fixed and forwarding untagged for all Ports of the Bridge. This entry may be modified or removed from the Permanent Database by means of the management operations defined in Clause 12 if the implementation supports these operations.

NOTE 2—This initial state causes the default tagging state for the PVID to be untagged, and for all other VIDs to be tagged, unless otherwise configured; however, the management configuration mechanisms allow any VID (including the PVID) to be specified as VLAN-tagged or untagged on any Port.

8.8.3 Dynamic Filtering Entries

A Dynamic Filtering Entry specifies

- a) An individual MAC Address;
- b) The FID, an identifier assigned by the MAC Bridge (8.8.8) to identify a set of VIDs for which no more than one Dynamic Filtering Entry can exist for any individual MAC Address;

NOTE 1—An FID identifies a set of VIDs among which *Shared VLAN Learning* (3.165) takes place. Any pair of FIDs identifies two sets of VIDs between which *Independent VLAN Learning* (3.75) takes place. The allocation of FIDs by a Bridge is described in 8.8.8.

- c) A Port Map specifying forwarding for the destination MAC Address and FID to a single Port.

NOTE 2—This is equivalent to specifying a single port number; hence, this specification is directly equivalent to the specification of dynamic entries in IEEE Std 802.1D, 1993 Edition [B9].

The Port Map may contain a **connection_identifier** (8.8.12) for each outbound Port.

Dynamic Filtering Entries are created and updated by the Learning Process (8.7). They shall be automatically removed after a specified time, the Ageing Time, has elapsed since the entry was created or last updated. No more than one Dynamic Filtering Entry shall be created in the Filtering Database for a given combination of MAC Address and FID.

Dynamic Filtering Entries cannot be created or updated by management.

NOTE 3—Dynamic Filtering Entries may be read by management (Clause 12). The FID is represented in the management Read operation by any one of the VIDs that it represents. For a given VID, the set of VIDs that share the same FID may also be determined by management.

The ageing out of Dynamic Filtering Entries ensures that end stations that have been moved to a different part of the network will not be permanently prevented from receiving frames. It also takes account of changes in the active topology of the network that can cause end stations to appear to move from the point of view of the bridge; i.e., the path to those end stations subsequently lies through a different Bridge Port.

The Ageing Time may be set by management (Clause 12). A range of applicable values and a recommended default is specified in Table 8-6; this is suggested to remove the need for explicit configuration in most cases. If the value of Ageing Time can be set by management, the Bridge shall have the capability to use values in the range specified, with a granularity of 1 s.

Table 8-6—Ageing time parameter value

Parameter	Recommended default value	Range
Ageing time	300.0 s	10.0–1 000 000.0 s

NOTE 4—The granularity is specified in order to establish a common basis for the granularity expressed in the management operations defined in Clause 12, not to constrain the granularity of the actual timer supported by a conformant implementation. If the implementation supports a granularity other than 1 s, then it is possible that the value read back by management following a Set operation will not match the actual value expressed in the Set.

The Spanning Tree Algorithm and Protocol specified in Clause 8 of IEEE Std 802.1D, 1998 Edition, includes a procedure for notifying all Bridges in the network of topology change. It specifies a short value

for the Ageing Timer, to be enforced for a period after any topology change (8.3.5 of IEEE Std 802.1D, 1998 Edition). While the topology is not changing, this procedure allows normal ageing to accommodate extended periods during which addressed end stations do not generate frames themselves, perhaps through being powered down, without sacrificing the ability of the network to continue to provide service after automatic configuration.

8.8.4 MAC Address Registration Entries

A MAC Address Registration Entry specifies

- a) A MAC Address specification, comprising
 - 1) An individual MAC address; or
 - 2) A group MAC address; or
 - 3) All Group Addresses, for which no more specific Static Filtering Entry exists; or
 - 4) All Unregistered Group Addresses, for which no more specific Static Filtering Entry exists.
- b) The VID for which the dynamic filtering information was registered;
- c) A Port Map, consisting of a control element for each outbound Port, which specifies forwarding (Registered) or filtering (Not registered) of frames destined to the MAC Address and VID.

MAC Address Registration Entries are created, modified, and deleted by the operation of MMRP (Clause 10). No more than one MAC Address Registration Entry shall be created in the Filtering Database for a given combination of MAC Address specification and VID.

NOTE—It is possible to have a Static Filtering Entry that has values of Forward or Filter on some or all Ports that mask the dynamic values held in a corresponding MAC Address Registration Entry. The values in the MAC Address Registration Entry will continue to be updated by MMRP; hence, subsequent modification of that entry to allow the use of dynamic filtering information on one or more Ports immediately activates the true MMRP registration state that was hitherto masked by the static information.

The creation of MAC Address Registration Entries is subject to the Restricted_MAC_Address_Registration management control (10.12.2.3). If the value of this control is TRUE, a dynamic entry for a given MAC address may only be created if a Static Filtering Entry already exists for that MAC address, in which the Registrar Administrative Control value is Normal Registration.

8.8.5 Dynamic VLAN Registration Entries

A Dynamic VLAN Registration Entry specifies

- a) The VID to which the dynamic filtering information applies;
- b) A Port Map with a control element for each outbound Port specifying whether the VID is registered on that Port.

A separate Dynamic VLAN Registration Entry with a distinct Port Map may be created for each VID from which frames are received by the Forwarding Process.

The creation of Dynamic VLAN Registration Entries is subject to the Restricted_VLAN_Registration management control (11.2.3.2.3). If the value of this control is TRUE, a dynamic entry for a given VID may only be created if a Static VLAN Registration Entry already exists for that VID, in which the Registrar Administrative Control value is Normal Registration.

8.8.6 Default Group filtering behavior

Forwarding and filtering of group-addressed frames may be managed by specifying defaults for each VID and outbound Port. The behavior of each of these defaults, as modified by the control elements of more

explicit Filtering Database entries applicable to a given frame's MAC Address, VID, and outbound Port, is as follows:

NOTE 1—As stated in 8.8.1, for a given MAC Address, there may be separate Static Filtering Entries with a distinct Port Map for each VID.

- a) *Forward All Groups.* The frame is forwarded, unless an explicit Static Filtering Entry specifies filtering independent of any dynamic filtering information.
- b) *Forward Unregistered Groups.* The frame is forwarded, unless
 - 1) An explicit Static Filtering Entry specifies filtering independent of any dynamic filtering information; or
 - 2) An explicit Static Filtering Entry specifies forwarding or filtering on the basis of dynamic filtering information, and an applicable explicit MAC Address Registration Entry exists specifying filtering; or
 - 3) An applicable explicit Static Filtering Entry does not exist, but an applicable MAC Address Registration Entry specifies filtering.
- c) *Filter Unregistered Groups.* The frame is filtered unless
 - 1) An explicit Static Filtering Entry specifies forwarding independent of any dynamic filtering information; or
 - 2) An explicit Static Filtering Entry specifies forwarding or filtering on the basis of dynamic filtering information, and an applicable explicit MAC Address Registration Entry exists specifying forwarding; or
 - 3) An applicable explicit Static Filtering Entry does not exist, but an applicable MAC Address Registration entry specifies forwarding.

In Bridges that support only Basic Filtering Services, the default Group filtering behavior is Forward All Groups for all Ports of the Bridge, for all VIDs.

NOTE 2—Forward All Groups corresponds directly to the behavior specified in IEEE Std 802.1D, 1993 Edition [B9] when forwarding group MAC Addressed frames for which no static filtering information exists in the Filtering Database. Forward All Groups makes use of information contained in Static Filtering Entries for specific group MAC Addresses, but overrides any information contained in MAC Address Registration Entries. Forward Unregistered Groups is analogous to the forwarding behavior of a Bridge with respect to individual MAC Addresses. If there is no static or dynamic information for a specific group MAC Address, then the frame is forwarded; otherwise, the frame is forwarded in accordance with the statically configured or dynamically learned information.

In Bridges that support Extended Filtering Services, the default Group filtering behavior for each outbound Port for each VID is determined by the following information contained in the Filtering Database:

- d) Any Static Filtering Entries applicable to that VID with a MAC Address specification of All Group Addresses or All Unregistered Group Addresses;
- e) Any MAC Address Registration Entries applicable to that VID with a MAC Address specification of All Group Addresses or All Unregistered Group Addresses.

The means whereby this information determines the default Group filtering behavior is specified in 8.8.9, Table 8-8, and Table 8-9.

NOTE 3—The result is that the default Group filtering behavior for each VID can be configured for each Port of the Bridge via Static Filtering Entries, determined dynamically via MAC Address Registration Entries created/updated by MMRP (Clause 10), or both. For example, in the absence of any static or dynamic information in the Filtering Database for All Group Addresses or All Unregistered Group Addresses, the default Group filtering behavior will be Filter Unregistered Groups on all Ports, for all VIDs. Subsequently, the creation of a Dynamic MAC Address Registration Entry for All Unregistered Group Addresses indicating “Registered” for a given VID on a given Port would cause that Port to exhibit Forward Unregistered Groups behavior for that VID. Similarly, creating a Static Filtering Entry for All Group Addresses indicating “Registration Fixed” on a given Port for that VID would cause that Port to exhibit Forward All Groups behavior.

Hence, by using appropriate combinations of “Registration Fixed,” “Registration Forbidden,” and “Normal Registration” in the Port Maps of Static Filtering Entries for the All Group Addresses and All Unregistered Group Addresses address specifications, it is possible, for a given Port and VID, to

- Fix the default Group filtering behavior to be just one of the three behaviors described above; or
- Restrict the choice of behaviors to a subset of the three, and allow MMRP registrations (or their absence) to determine the final choice; or
- Allow any one of the three behaviors to be adopted, in accordance with any registrations received via MMRP.

8.8.7 Dynamic Reservation Entries

A Dynamic Reservation Entry specifies the following:

- a) A group MAC address or an individual MAC address.
- b) The VID for which the dynamic reservation information was registered.
- c) A Reservation Port Map, consisting of a control element for each outbound Port that specifies forwarding (reservation information exists for this Port) or filtering (reservation information does not exist for this Port) of frames for this Port. The initial value shall be filtering.

Dynamic Reservation Entries are created, modified, and deleted by the operation of SRP. No more than one Dynamic Reservation Entry shall be created in the Filtering Database for a given combination of MAC Address and VID.

8.8.8 Allocation of VIDs to FIDs

The allocation of VIDs to FIDs within a Bridge determines how learned individual MAC Address information is used in forwarding/filtering decisions within a Bridge; whether such learned information is confined to individual VIDs, shared among all VIDs, or confined to specific sets of VIDs.

The allocation of VIDs to FIDs is determined on the basis of

- a) The set of VLAN Learning Constraints that have been configured into the Bridge (by means of the management operations defined in Clause 12);
- b) Any fixed mappings of VIDs to FIDs that may have been configured into the Bridge (by means of the management operations defined in Clause 12);
- c) The *set of active VIDs* (i.e., those VIDs on whose behalf the Bridge may be called on to forward frames). A VID is active if either of the following is true:
 - 1) The VID’s member set (8.8.10) contains one Port that is in a forwarding state, and at least one other Port of the Bridge is both in a forwarding state and has Ingress Filtering (8.6.2) disabled;
 - 2) The VID’s member set contains two or more Ports that are in a forwarding state.
- d) The capabilities of the Bridge with respect to the number of FIDs that it can support, and the number of VIDs that can be allocated to each FID.

A Bridge shall support a minimum of one FID and may support up to 4094 FIDs. For the purposes of the management operations, FIDs are numbered from 1 through N, where N is the maximum number of FIDs supported by the implementation.

A Bridge shall support the ability to allocate at least one VID to each FID and may support the ability to allocate more than one VID to each FID.

The number of VLAN Learning Constraints supported by a Bridge is an implementation option.

NOTE—In an SVL/IVL Bridge (3.167), a number of FIDs are supported, and one or more VIDs can be mapped to each FID. In an SVL Bridge (3.166), a single FID is supported, and all VIDs are mapped to that FID. In an IVL Bridge (3.76), a number of FIDs are supported, and only one VID can be mapped to each FID.

An MST Bridge shall support the ability to allocate at least one FID to each spanning tree and may support the ability to allocate more than one FID to each spanning tree.

NOTE—In other words, the number of FIDs supported by the Bridge must be greater than or equal to the number of spanning trees supported by the Bridge.

An MST Bridge shall ensure that the maximum supported numbers of FIDs and VIDs can be associated unambiguously. This requires either 1) a number of fixed VID to FID allocations at least equal to the maximum number of VIDs supported; or 2) one I Constraint entry per FID supported and one S Constraint entry per MSTI supported, or both (8.8.8.2).

8.8.8.1 Fixed and dynamic VID to FID allocations

A Bridge may support the ability to define fixed allocations of specific VIDs to specific FIDs, via an allocation table that may be read and modified by means of the management operations defined in Clause 12. For each VID supported by the implementation, the allocation table indicates one of the following:

- a) The VID is currently not allocated to any FID; or
- b) A fixed allocation has been defined (via management), allocating this VID to a specific FID; or
- c) A dynamic allocation has been defined (as a result of applying the VLAN Learning Constraints), allocating this VID to a specific FID.

For any VID that has no fixed allocation defined, the Bridge can dynamically allocate that VID to an appropriate FID, in accordance with the current set of VLAN Learning Constraints.

8.8.8.2 VLAN Learning Constraints

There are two types of VLAN Learning Constraint:

- a) A Shared Learning Constraint (or S Constraint) asserts that Shared VLAN Learning shall occur between a pair of identified VIDs. S Constraints are of the form {A S B}, where A and B are VIDs. An S Constraint is interpreted as meaning that Shared VLAN Learning shall occur between the pair of VIDs;
- b) An Independent Learning Constraint (or I Constraint) asserts that a given VID is a member of a set of VIDs among which Independent VLAN Learning shall occur. I Constraints are of the form {A I N}, where A is a VID and N is an Independent Set Identifier. An I Constraint is interpreted as meaning that Independent VLAN Learning shall occur among the set of VIDs comprising VID A and all other VIDs identified in I Constraints that carry the same Independent Set Identifier, N.

A given VID may appear in any number (including zero) of S Constraints and/or I Constraints.

NOTE 1—S Constraints are

- *Symmetric*: e.g., {A S B} and {B S A} both express an identical constraint;
- *Transitive*: e.g., {A S B}, {B S C} implies the existence of a third constraint, {A S C};
- *Reflexive*: e.g., {A S A} is a valid S Constraint.

I Constraints are not

- *Symmetric*: e.g., {A I 1} is a valid constraint and {1 I A} is not well formed;
- *Transitive*: e.g., ({A I 1}, {B I 1}, {B I 2}, {C I 2}) does not imply either {A I 2} or {C I 1}.

The allocation of VIDs to FIDs shall be such that, for all members of the set of active VIDs (8.8.8),

- c) A given VID shall be allocated to exactly one FID;
- d) If a given VID appears in an I Constraint, then it shall not be allocated to the same FID as any other VID that appears in an I Constraint with the same Independent Set Identifier;
- e) If a given VID appears in an S Constraint (either explicit, or implied by the transitive nature of the specification), then it shall be allocated to the same FID as the other VID identified in the same S Constraint;
- f) If a VID does not appear in any S or I Constraints, then the Bridge may allocate that VID to any FID of its choice.

NOTE 2—The intent is that the set of Learning Constraints is defined on a global basis; i.e., that all VLAN-aware Bridges are configured with the same set of constraints (although individual constraints may well be defined and distributed by different managers/administrators). Any Bridge, therefore, sees the complete picture in terms of the Learning Constraints that apply to all VIDs present in the network, regardless of whether they all apply to VIDs that are present in that particular Bridge. This standard provides the definition, in Clause 12, of managed objects and operations that model how individual constraints can be configured in a Bridge; however, the issue of how a distributed management system might ensure the consistent setting of constraints in all Bridges in a network is not addressed by this standard.

8.8.8.3 VLAN Learning Constraint inconsistencies and violations

The application of the rules specified in 8.8.8.2, coupled with any fixed allocations of VIDs to FIDs that may have been configured, can result in the Bridge detecting Learning Constraint inconsistencies and/or violations (i.e., can result in situations where there are inherent contradictions in the combined specification of the VLAN Learning Constraints and the fixed allocations, or the Bridge's own limitations mean that it cannot meet the set of VLAN Learning Constraints that have been imposed on it).

A Bridge detects a Learning Constraint inconsistency if

- a) The VLAN Learning Constraints, coupled with any fixed VID to FID allocations, are such that, if any given pair of VIDs became members of the set of active VIDs (8.8.8), the result would be a simultaneous requirement for Independent VLAN Learning and for Shared VLAN Learning for those two VIDs. Such an inconsistency would require the Bridge to allocate that pair of VIDs both to the same FID and to different FIDs.

Learning Constraint inconsistencies are detected when a management operation (12.10.3) attempts to set a new Learning Constraint value, or to modify the fixed VID to FID allocations. If the new constraint or allocation that is the subject of the operation is inconsistent with those already configured in the Bridge, then the management operation shall not be performed and an error response shall be returned.

A Bridge detects a Learning Constraint violation if

- b) The Bridge does not support the ability to map more than one VID to any given FID, and the VLAN Learning Constraints indicate that two or more members of the active set of VIDs require to be mapped to the same FID; or
- c) The number of FIDs required in order to correctly configure the Bridge to meet the VLAN Learning Constraints and fixed VID to FID allocations for all members of the active set of VIDs exceeds the number of FIDs supported by the Bridge.

Learning Constraint violations are detected

- d) When a VID that was hitherto not a member of the set of active VIDs (8.8.8) becomes active, either as a result of management action or as a result of the operation of MVRP, resulting in the Bridge no

longer being able to support the defined set of constraints and/or fixed allocations for the set of active VIDs; or

- e) When other management reconfiguration actions, such as defining a new Learning Constraint or fixed VID to FID allocation, results in the Bridge no longer being able to support the defined set of constraints and/or fixed allocations for the set of active VIDs.

On detection of a violation, the Bridge issues the Notify Learning Constraint Violation management notification (12.11) in order to alert any management stations to the existence of the violation. There is the potential for a single change in configuration to result in more than one VID whose constraints cannot be met; in such cases, multiple notifications are generated.

8.8.9 Querying the Filtering Database

If the VID assigned to a relayed frame identifies a VID or an ESP with a given outbound Bridge Port in its member set (8.8.10), the Filtering Database entries that are applicable to the frame's destination MAC Address and VID determine whether the frame is filtered or forwarded through that Bridge Port. More than one entry can apply to a given frame. This subclause (8.8.9) specifies the effect of combining applicable entries, and of the absence of certain types of entries (not all possible entries are necessarily present). For an overview of the different types of entries, see the introductory text of 8.8.

The Filtering Database entries applicable to a frame whose destination MAC Address is a specific Individual MAC Address are the Dynamic Filtering Entry (if present) with that MAC Address and the FID to which the frame's VID is allocated (8.8.8), and all Static Filtering Entries (if any) with that MAC Address, or with "All Individual MAC Address," as their MAC Address specification and a VID that is also allocated to that FID, as their VLAN identifier specification. An entry with a wildcard VID applies only if there is no applicable Static Filtering Entry for a specific VID. Table 8-7 specifies the result of combining this information for an individual MAC address and a given outbound Bridge Port, and can be summarized as follows:

- IF any static entry for a VID allocated to the FID specifies Forward THEN Forward
- ELSE IF any static entry for a VID allocated to the FID specifies Filter THEN Filter
- ELSE IF a static entry for the wildcard VID specifies Forward THEN Forward
- ELSE IF a static entry for the wildcard VID specifies Filter THEN Filter
- ELSE IF dynamic (learnt filtering information) entry for the FID specifies Filter THEN Filter
- ELSE Forward

Table 8-7—Combining Static and Dynamic Filtering Entries for an individual MAC Address

		Control Elements in any Static Filtering Entry or Entries for this individual MAC Address, FID, and outbound Port specify:			
Filtering Information	Forward (Any Static Filtering Entry for this Address/FID/Port specifies Forward)	Filter (No Static Filtering Entry for this Address/FID/Port specifies Forward)	Use Dynamic Filtering Information (No Static Filtering Entry for this Address/FID/Port specifies Forward or Filter), or no Static Filtering Entry present. Dynamic Filtering Entry Control Element for this individual MAC Address, FID and outbound Port specifies:		
	Forward	Filter	Forward	Filter	No Dynamic Filtering Entry present
Result	Forward	Filter	Forward	Filter	Forward

The Filtering Database entries applicable (if present) to a frame whose destination MAC Address is a specific group MAC Address are the Static Filtering Entries and MAC Address Registration Entries (for the frame's VID and for the wildcard VID) whose address specification is that group MAC address or "All Group Addresses" or "All Unregistered Group Addresses." A Static Filtering Entry for a wildcard VID applies only if there is no applicable Static Filtering Entry for a specific VID, MAC Address Registration Entries do not use wildcard VIDs. Table 8-8 specifies the results, Registered or Not Registered for a given outbound Bridge Port, of combining the entries for the "All Group Addresses" and for combining the entries for "All Unregistered Group Addresses." Table 8-9 combines these results with the entries for the specific group MAC Address. The result of Table 8-8 for "All Group Addresses" can be summarized as follows:

- IF a static entry for "All Group Addresses" and the frame's VID specifies Forward (Registration Fixed) THEN "All Group Addresses" is Registered
- ELSE IF a static entry for "All Group Addresses" and the frame's VID specifies Filter (Registration Forbidden) THEN "All Group Addresses" is Not Registered
- ELSE IF a dynamic (MAC Address Registration) entry for "All Group Addresses" and the frame's VID specifies Forward (Registered) THEN "All Group Addresses" is Registered
- ELSE "All Group Addresses" is NOT Registered

NOTE 1—The result for "All Unregistered Group Addresses" is given by substituting that address specification for "All Group Address" throughout the summary.

Table 8-8—Combining Static Filtering Entry and MAC Address Registration Entry for "All Group Addresses" and "All Unregistered Group Addresses"

		Static Filtering Entry Control Element for this group MAC Address, VID, and outbound Port specifies:			
Filtering Information	Registration Fixed (Forward)	Registration Forbidden (Filter)	Use MAC Address Registration Information, or no Static Filtering Entry present. MAC Address Registration Entry Control Element for this group MAC Address, VID and outbound Port specifies:		
			Registered (Forward)	Not Registered (Filter)	No MAC Address Registration Entry present
Result	Registered	Not Registered	Registered	Not Registered	Not Registered

The result of Table 8-9 can be summarized as follows:

- IF a static entry for the specific group address and the frame's VID specifies Forward THEN Forward
- ELSE IF a static entry for the specific group address and the frame's VID specifies Filter THEN Filter
- ELSE IF a static entry for the specific group address and the wildcard VID specifies Forward THEN Forward
- ELSE IF a static entry for the specific group address and the wildcard VID specifies Filter THEN Filter
- ELSE IF the Table 8-8 result for "All Group Addresses" is Registered THEN Forward
- ELSE IF the Table 8-8 result for "All Unregistered Group Addresses" is Registered THEN Forward

- ELSE IF a dynamic (MAC Address Registration) entry for the specific group address and the frame's VID specifies Forward THEN Forward
- ELSE Filter

Table 8-9—Forwarding or Filtering for specific group MAC Addresses

		Static Filtering Entry Control Element for this group MAC Address, VID and outbound Port specifies:				
		Registration Fixed (Forward)	Registration Forbidden (Filter)	Use MAC Address Registration Information, or no Static Filtering Entry present. MAC Address Registration Entry Control Element for this group MAC Address, VID and outbound Port specifies:		
				Registered (Forward)	Not Registered (Filter)	No MAC Address Registration Entry present
All Group Addresses control elements for this VID and Port specify (Table 8-8):	Not Registered	Forward	Filter	Forward	Filter (Filter Unregistered Groups)	Filter (Filter Unregistered Groups)
	Registered	Forward	Filter	Forward	Forward (Forward Unregistered Groups)	Forward (Forward Unregistered Groups)
	Registered	Forward	Filter	Forward (Forward All Groups)	Forward (Forward All Groups)	Forward (Forward All Groups)

When forwarding or filtering a frame with a destination group MAC Address, a VLAN-aware Bridge may:

- a) Ignore the allocation of VIDs to FID, and use Table 8-9 directly for the frame's VID; or
- b) Take the same decision for all VIDs allocated to any given FID, forwarding if Table 8-9 specifies Forward for any VID allocated to the same FID as the frame's VID, and filtering otherwise.

For a given destination MAC address and VID, a Dynamic Reservation Entry (8.8.7) may be present in the filtering database, indicating, by means of its port map, which Ports have valid bandwidth reservations. If a Dynamic Reservation Entry exists in the filtering database for that MAC address and VID, the port map in the Dynamic Reservation Entry is used to prevent frames being forwarded on Ports where no reservation exists. Table 8-10 combines the output of Table 8-7 (for an individual MAC address) or Table 8-9 (for a group MAC address) with a Dynamic Reservation Entry to specify forwarding, or filtering, of a frame with that destination MAC Address and VID through an outbound Port.

Table 8-10—Forwarding or Filtering with Dynamic Reservation Entries

Output of Table 8-7 (individual MAC address) or Table 8-9 (group MAC address) for this group or individual MAC Address, VID and outbound Port specifies:	Dynamic Reservation Entry Control Element for this individual or group MAC Address, VID and outbound Port specifies:	
	Forward	Filter
Forward	Forward	Filter
Filter	Filter	Filter

NOTE 2—Where a group MAC address is used as the destination address for reserved traffic, the reservation mechanisms used by SRP will prevent reservations being made for multiple streams using the same combination of MAC address and VID. The Dynamic Reservation Entry can then control which Port(s) the stream associated with the MAC address and VID is permitted to be transmitted through. The restriction preventing multiple streams from using the same group MAC address/VID is necessary because those different streams may be destined for different destination Ports; if the restriction was not imposed, it would not be possible to distinguish which frames belong to which streams, and therefore, through which Ports they should be transmitted.

8.8.10 Determination of the member set for a VID

The VLAN configuration information contained in the Filtering Database for a given VID may include a Static VLAN Registration Entry (8.8.2) and/or a Dynamic VLAN Registration Entry (8.8.5). This information defines the member set, the Ports through which members of the VLAN identified by that VID can be reached.

For a given VID, the Filtering Database can contain a Static VLAN Registration Entry, a Dynamic VLAN Registration Entry, both, or neither. Table 8-11 combines Static VLAN Registration Entry and Dynamic VLAN Registration Entry information for a VID and Port to give a result *member*, or *not member*, for the Port. The member set for a given VID consists of all Ports for which the result is member.

Table 8-11—Determination of whether a Port is in a VID's member set

Filtering Information	Static VLAN Registration Entry Control Element for this VID and Port specifies:					
	Registration Fixed	Registration Forbidden	Normal Registration, or no Static VLAN Registration Entry present. Dynamic VLAN Registration Entry Control Element for this VID and Port specifies:			
			Registered	Not Registered	No Dynamic VLAN Registration Entry present	
Result	Member	Not member	Member	Not member	No Dynamic VLAN Registration Entry present	Not member

The Forwarding Process uses the member set to apply ingress (8.6.2) and egress rules (8.6.4) for the VID.

8.8.11 Permanent Database

The Permanent Database provides fixed storage for a number of Static Filtering Entries and Static VLAN Registration Entries. The Filtering Database shall be initialized with the Filtering Database Entries contained in this fixed data store.

Entries may be added to and removed from the Permanent Database under explicit management control, using the management functionality defined in Clause 12. Changes to the contents of Static Filtering Entries or Static VLAN Registration Entries in the Permanent Database do not affect forwarding and filtering decisions taken by the Forwarding Process or the egress rules until such a time as the Filtering Database is reinitialized.

NOTE 1—This aspect of the Permanent Database can be viewed as providing a “boot image” for the Filtering Database, defining the contents of all initial entries, before any dynamic filtering information is added.

NOTE 2—Subclause 10.12.2.3 defines an initial state for the contents of the Permanent Database, required for the purposes of MMRP operation.

8.8.12 Connection_Identifier

A connection_identifier may be associated with the Bridge Port value maintained in a Dynamic Filtering Entry of the Filtering Database for Bridge Ports that generate EM_UNITDATA.indications with a non-null connection_identifier parameter (e.g., a Virtual Instance Port as specified in 6.10). The connection_identifier has no effect on the Bridge operation specified in this clause other than being stored in the Filtering Database and conveyed across the EISS as a parameter in an EM_UNITDATA.request or EM_UNITDATA.indication (6.8.1).

When the result of a Filtering Database query (8.8.9) for a particular frame is that the frame will be forwarded, and the Filtering Database entry determining that result has a connection_identifier, then that connection_identifier is included in the EM_UNITDATA.request. For a given VID and MAC address, a connection_identifier can be included in a Dynamic Filtering Entry of the Filtering Database. The Filtering Database entry that determines a filtering or forwarding result is specified in Table 8-7. If there are no entries for that MAC address and VID in the Filtering Database, then a null value for the connection_identifier is used in the EM_UNITDATA.request.

When an EM_UNITDATA.indication with a non-null connection_identifier is received from an ingress port, and the Learning Process identifies or creates a Dynamic Filtering Entry for that frame’s source address and VID, the connection_identifier is included in the entry created for that port in the Filtering Database.

8.9 MST and ESP configuration information

In order to support multiple spanning trees, an MST Bridge has to be configured with an unambiguous assignment of VIDs to spanning trees. This is achieved by:

- a) Ensuring that the allocation of VIDs to FIDs (8.8.8) is unambiguous; and
- b) Ensuring that each FID supported by the Bridge is allocated to exactly one Spanning Tree.

The first of these requirements is met by configuring a set of VLAN learning constraints and/or fixed VID to FID mappings that are self-consistent, and which define an I Constraint, an S Constraint, or a fixed VID to FID allocation for all VIDs supported by the Bridge.

The second requirement is met by means of the FID to MSTI Allocation Table (8.9.3).

The combination of the VID to FID allocations and the FID to MSTI allocations defines a mapping of VIDs to spanning trees, represented by the MST Configuration Table (8.9.1).

A Bridge supporting PBB-TE can assign specific VID values to ESPs but can also support any of the spanning tree protocols. This is achieved by allocating the VIDs associated with the PBB-TE ESPs to a special MSTID, the TE-MSTID.

8.9.1 MST Configuration Table

The MST Configuration Table defines, for each VID, the MSTID of the spanning tree instance to which the VID is allocated.

In a Bridge that supports PBB-TE an MSTID of 0xFFE, the TE-MSTID, identifies VIDs that can be used by ESPs. Each of these VIDs, the ESP-VIDs, shall be allocated to an individual FID.

The MST Configuration Table cannot be configured directly; configuration of the table occurs as a consequence of configuring the relationships between VIDs and FIDs (8.8.8) and between FIDs and spanning trees (8.9.3).

8.9.2 MST configuration identification

For two or more MST Bridges to be members of the same MST Region (3.113), it is necessary for those Bridges to be directly connected together (i.e., interconnected only by means of LANs, without intervening Bridges that are not members of the region), and for them to support the same MST Region configuration. Two MST Region configurations are considered to be the same if the correspondence between VIDs and spanning trees is identical in both configurations.

NOTE 1—If two adjacent MST Bridges consider themselves to be in the same MST Region despite having different mappings of VIDs to spanning trees, then the possibility exists of undetectable loops arising within the MST Region.

In order to ensure that adjacent MST Bridges are able to determine whether they are part of the same MST Region, the MST BPDU supports the communication of an MST Configuration Identifier (13.7).

NOTE 2—As the MST Configuration Identifier is smaller than the mapping information that it summarizes, there is a small but finite possibility that two MST Bridges will assume that they have the same MST Region Configuration when this is not actually the case. However, given the size of the identifier, this standard assumes that this possibility is sufficiently small that it can safely be ignored. Appropriate use of the Configuration Name and Revision Level portions of the identifier can remove the possibility of an accidental match between MST Configuration Identifiers that are derived from different configurations within a single administrative domain (see 13.7).

8.9.3 FID to MSTI Allocation Table

The FID to MSTI Allocation Table defines, for all FIDs that the Bridge supports, the MSTID of the spanning tree instance to which the FID is allocated. An MSTID of zero is used to identify the CIST.

NOTE—The management operations defined in 12.12 make use of the concept of an MSTI List to instantiate/de-instantiate MST instances and will only permit the allocation of FIDs to MSTIDs that are present in the MSTI List.

8.10 Spanning Tree Protocol Entity

Figure 8-5 illustrates the operation of the Spanning Tree Protocol Entity, including the reception and transmission of frames containing BPDUs, the modification of the state information associated with individual Bridge Ports, and the notification of the Filtering Database of changes in active topology.

A given MST bridge is not required to support all of the spanning trees that exist within the MST bridged network. That is, the number of spanning trees operated by the Spanning Tree Protocol Entity in a given bridge may be different from the number operated by that in another bridge. However, as a direct consequence of the conditions stated in 8.9.2, the number of instances of the Spanning Tree Protocol operated by a given MST Bridge is the same for all Bridges that are members of a given MST Region.

8.11 MRP Entities

The MRP entities operate the Algorithms and Protocols associated with the MRP Applications supported by the Bridge and consist of the set of MRP Participants for those MRP Applications (Clause 10).

Figure 8-6 illustrates the operation of an MRP entity, including the reception and transmission of frames containing MRPDUs, the use of control information contained in the Filtering Database, and the notification of the Filtering Database of changes in filtering information.

8.12 Bridge Management Entity

In order to facilitate interoperable management of Bridges by remote means (as opposed to management via some kind of management console attached directly to the Bridge), it is recommended that SNMP should be the protocol that is used, in conjunction with the SMIv2 MIB modules specified in Clause 17.

8.13 Addressing

All MAC Entities communicating across a Bridged Local Area Network use EUI-48 addresses. These can be Universally Administered Addresses or a combination of Universally Administered and Locally Administered Addresses.

8.13.1 End stations

Frames transmitted between end stations using the MAC Service carry the MAC Address of the source and destination peer end stations in the source and destination address fields of the frames, respectively. The address, or other means of identification, of a Bridge is not carried in frames transmitted between peer users for the purpose of frame relay in the network.

In the absence of explicit filters configured via management as Static Filtering Entries, or via MMRP as MAC Address Registration Entries (8.8, Clause 10), frames with a destination address of the broadcast address or any other group address that is not a Reserved Address (8.6.3) are assigned a VID and relayed throughout the VLAN identified by that VID.

8.13.2 Bridge Ports

A separate individual MAC Address is associated with each instance of the MAC Service provided to an LLC Entity. That MAC Address is used as the source address of frames transmitted by the LLC Entity.

Media access method specific procedures can require the transmission and reception of frames that use an individual MAC Address associated with the Bridge Port, but neither originate from nor are delivered to a MAC Service user. Where an individual MAC Address is associated with the provision of an instance of the MAC Service by the Port, that address can be used as the source and/or the destination address of such frames, unless the specification of the media access method specific procedures requires otherwise.

8.13.3 Use of LLC by Spanning Tree Protocol Entities

Spanning Tree Protocol Entities use the DL_UNITDATA.request and DL_UNITDATA.indication primitives (ISO/IEC 8802-2) provided by individual LLC Entities associated with each Bridge Port to transmit and receive frames. The source_address and destination_address parameters of the DL_UNITDATA.request shall both denote the standard LLC address assigned to the Bridge Spanning Tree Protocol (Table 8-12).

Each DL_UNITDATA request primitive gives rise to the transmission of an LLC UI command PDU, which conveys the BPDU in its information field.²⁴

Table 8-12—Standard LLC address assignment

Assignment	Value
Bridge spanning tree protocol	01000010

Code Representation: The least significant bit (LSB) of the value shown is the right-most. The bits increase in significance from right to left. It should be noted that the code representation used here has been chosen in order to maintain consistency with the representation used elsewhere in this standard, and with the representation used in IEEE Std 802.1D.

IEEE Std 802.1D defines a Protocol Identifier field, present in all BPDUs (Clause 14), which serves to identify different protocols supported within the scope of the LLC address assignment. Further values of this field are reserved for future standardization. A Spanning Tree Protocol Entity that receives a BPDU with an unknown Protocol Identifier shall discard that PDU.

8.13.4 Reserved MAC Addresses

Any frame with a destination address that is a Reserved MAC Address shall not be forwarded by a Bridge. Reserved MAC Addresses for C-VLAN components, for S-VLAN components, and for TPMR components are specified in Table 8-1, Table 8-2, and Table 8-3, respectively. These group MAC Addresses are reserved for assignment to standard protocols, according to the criteria for such assignments.²⁵

8.13.5 Group MAC Addresses for spanning tree protocols

A Spanning Tree Protocol Entity uses an instance of the MAC Service provided by each of the Bridge's Ports to transmit frames addressed to the Spanning Tree Protocol Entities of all the other Bridges attached to the same LAN as that Port. An EUI-48 universally administered Group Address, known as the Bridge Group Address, has been assigned (Table 8-1) for use by C-VLAN components for this purpose.

It is essential to the operation of the spanning tree protocols that frames with this destination address both reach all peer protocol entities attached to the same LAN and do not reach protocol entities attached to other LANs. The Bridge Group Address is therefore included in the block of C-VLAN components Reserved MAC Addresses and is always filtered by Customer Bridges and C-VLAN components of Provider Edge Bridges (8.6.3).

As a network of Provider Bridges needs to appear to attached Customer Bridges as if it was a single LAN, frames addressed to the Bridge Group Address are forwarded by S-VLAN components. Provider Bridges also use a Spanning Tree Protocol to provide one or more loop-free active topologies, so a distinct EUI-48 universally administered Group Address, the Provider Bridge Group Address, which can be confined to the LANs that their Bridge Ports attach, has been assigned (Table 8-2). The Provider Bridge Group Address is included in both the C-VLAN component and the S-VLAN component Reserved MAC Addresses and is always filtered by all Bridges (8.6.3).

²⁴Administration of LLC address assignments is the responsibility of the IEEE Registration Authority (IEEE RA). A full list of standard LLC address assignments, and the criteria for assignment, can be found on the IEEE RA website at <http://standards.ieee.org/regauth/index.html>.

²⁵Administration of standard Group MAC address assignments is the responsibility of the IEEE Registration Authority (IEEE RA). A full list of standard Group MAC address assignments, and the criteria for assignment, can be found on the IEEE RA website at <http://standards.ieee.org/regauth/index.html>.

The source MAC address field of frames transmitted by Spanning Tree Protocol Entities contains the individual MAC Address for the Bridge Port used to transmit the frame.

8.13.6 Group MAC Addresses for MRP Applications

An MRP Entity that supports a given MRP Application transmits frames addressed to all other MRP Entities that implement the same MRP Application. The peers of each such entity bound a region of the network that contains no peers, commonly a single LAN in the case where all Bridges attached to the LAN implement the application.

Universally administered EUI-48 Group Addresses are assigned to MRP applications, from a set of EUI-48 Universal Addresses, known as MRP application addresses, as shown in Table 10-1; these addresses are for use by VLAN components that support the MRP applications concerned. Filtering Database Entries for each MRP application address assigned to an application that is supported by a VLAN component should be configured in the Filtering Database so as to confine frames for that application to the peer region, while addresses for applications that are not supported should not be included. These group MAC addresses are reserved for assignment to standard protocols, according to the criteria for such assignments (see 8.13.4).

When an MRP Application operates among both Customer and Provider Bridges, a distinct EUI-48 universally administered Group Address is assigned for use by both C-VLAN components and S-VLAN components. MMRP is such an application, and uses the Customer and Provider Bridge MMRP Address assigned in Table 10-1. When an MRP application operates separately among Customer Bridges or among Provider Bridges, different distinct EUI-48 universally administered Group Addresses are assigned for use by C-VLAN components and for use by S-VLAN components. MVRP is such an application; C-VLAN components use the Customer Bridge MVRP address assigned in Table 10-1, while S-VLAN components use the Provider Bridge MVRP address assigned in Table 8-1. Note that because the Provider Bridge MVRP address is selected from the set of addresses that appear in Table 8-1 but not Table 8-2, it will always be filtered by C-VLAN components. Although the Provider Bridge MVRP Address is not selected from Table 10-1, an S-VLAN component should treat this as an MRP Application Address and configure a Filtering Database Entry to filter frames with this address.

An MRP Entity that supports a given MRP Application uses the individual MAC address for the Bridge Port through which the PDU is transmitted (8.13.2) as the source address field of MAC frames conveying MRPDUs for that application.

8.13.7 Bridge Management Entities

The recommended protocol for remote Bridge management is SNMP, which typically uses IP as a network layer protocol; however, SNMP can also be supported directly over an IEEE 802 LAN, as specified in IETF RFC 4789, Simple Network Management Protocol (SNMP) over IEEE 802 Networks. If implemented, the management protocol stack and address used shall be supported by a single LLC Entity attached to a Bridge Port. The Port should be a Management Port for the Bridge, as described in 8.3 and Figure 8-7, but may be a Port attached to a LAN, as described in 8.13.9, Figure 8-13, and Figure 8-14.

NOTE—An EUI-48 universally administered Group Address, known as the All LANs Bridge Management Group Address with a value of 01-80-C2-00-00-10 was assigned and recorded in the 1990 Edition of this standard. That address should not be used for Bridge management or for any other purpose.

8.13.8 Unique identification of a Bridge

A unique EUI-48 Universally Administered MAC Address, termed the Bridge Address, shall be assigned to each Bridge. The Bridge Address may be the individual MAC Address of a Bridge Port; in which case, use of the address of the lowest numbered Bridge Port (Port 1) is recommended.

NOTE—The Rapid Spanning Tree Protocol (RSTP) (Clause 13) and the Multiple Spanning Tree Protocol (MSTP) (Clause 13) require that a single unique identifier be associated with each Bridge. That identifier is derived from the Bridge Address as specified in 14.2.2.

8.13.9 Points of attachment and connectivity for Higher Layer Entities

The Higher Layer Entities in a Bridge, such as the Spanning Tree Protocol Entity (8.10), MRP entities (8.11), and Bridge Management (8.12), are modeled as attaching directly to one or more individual LANs connected by the Bridge's Ports, in the same way that any distinct end station is attached to the network. While these entities and the relay function of the Bridge use the same individual MAC entities to transmit and receive frames, the addressing and connectivity to and from these entities is the same as if they were attached as separate end stations “outside” the Port or Ports where they are actually attached. Figure 8-11 is functionally equivalent to Figure 8-2 but illustrates this logical separation between the points of attachment used by the Higher Layer Entities and those used by the MAC Relay Entity.

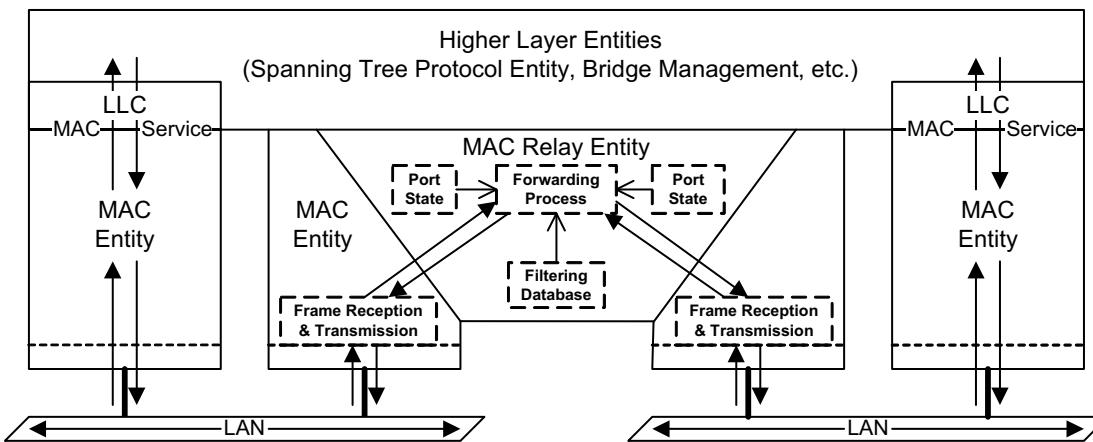


Figure 8-11—Logical points of attachment of the Higher Layer and Relay Entities

Figure 8-12 depicts the information used to control the forwarding of frames from one Bridge Port to another (the Port States and the content of the Filtering Database) as a series of switches (shown in the open, disconnected state) inserted in the path provided by the MAC Relay Entity. For the Bridge to forward a given frame between two Ports, all three switches must be in the closed state. While showing Higher Layer Entities sharing the point of attachment to each LAN used by each Bridge Port to forward frames, this figure further illustrates a point made by Figure 8-11. Controls placed in the forwarding path have no effect on the ability of a Higher Layer Entity to transmit and receive frames to or from a given LAN using a direct attachment to that LAN (e.g., from entity A to LAN A); they only affect the path taken by any indirect transmission or reception (e.g., from entity A to or from LAN B).

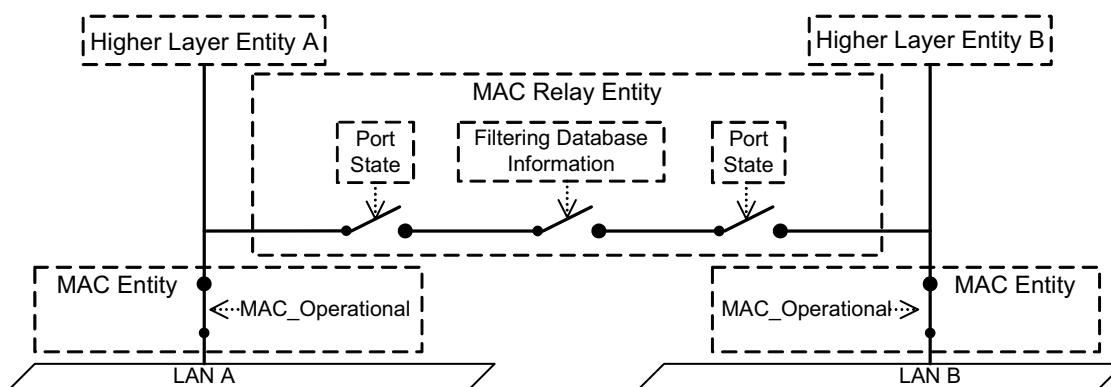


Figure 8-12—Effect of control information on the forwarding path

The functions provided by Higher Layer Entities can be categorized as requiring either

- a) A single point of attachment to the Bridged Local Area Network, providing connectivity to stations attached to the network at any point (subject to administrative control), as does Bridge Management; or
- b) A distinct point of attachment to each individual LAN attached by a Bridge Port, providing connectivity only to peer entities connected directly to that LAN, as do the Spanning Tree Protocol Entity and the MRP entity.

In the latter case, it is essential that the function associate each received and transmitted frame with a point of attachment. Frames transmitted or received via one point of attachment are not to be relayed to and from other Ports and attached LANs, so the MAC Addresses (8.13.4) used to reach these functions are permanently configured in the Filtering Database.

NOTE 1—Addresses used to reach functions with distinct points of attachment are generally group MAC Addresses.

NOTE 2—A single higher layer entity can incorporate both a function requiring a single point of attachment and a function requiring distinct points of attachment. The two functions are reached using different MAC addresses.

Figure 8-13 illustrates forwarding path connectivity for frames destined for Higher Layer Entities requiring per-Port points of attachment. Configuration of the Permanent Database in all Bridges to prevent relay of frames addressed to these entities means that they receive frames only via their direct points of attachment (i.e., from LAN A to entity A, and from LAN B to entity B), regardless of Port states.

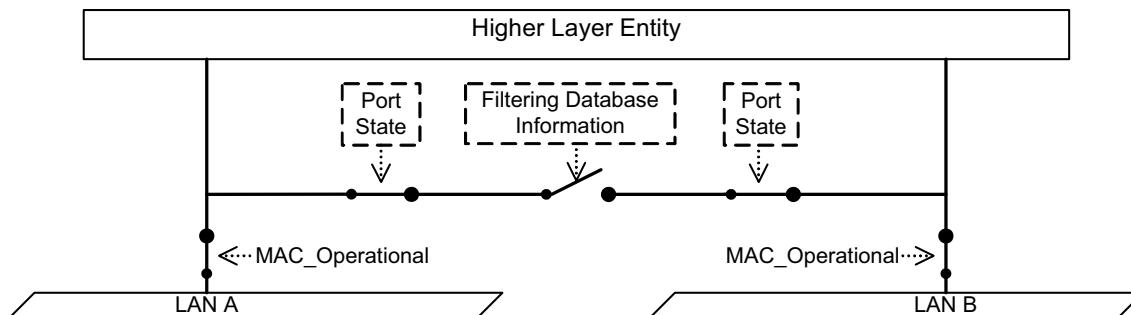


Figure 8-13—Per-Port points of attachment

Figure 8-14 and Figure 8-15 illustrate forwarding path connectivity for frames destined for a Higher Layer Entity requiring a single point of attachment. In both figures, the Filtering Database permits relay of frames, as do the Port states in Figure 8-14 where frames received from LAN B are relayed by the Bridge to the entity and to LAN A.

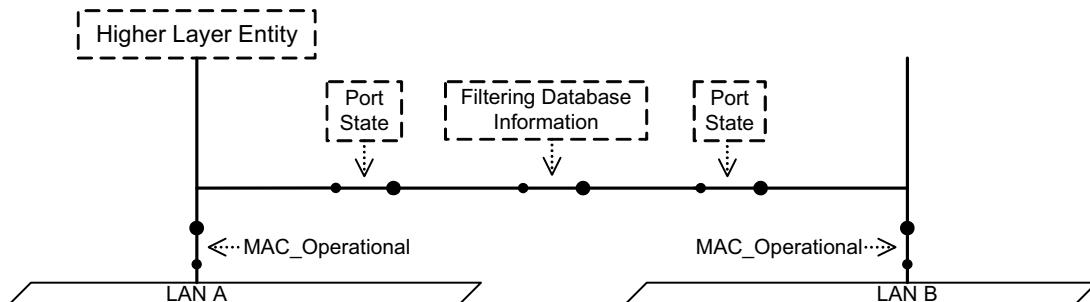


Figure 8-14—Single point of attachment—relay permitted

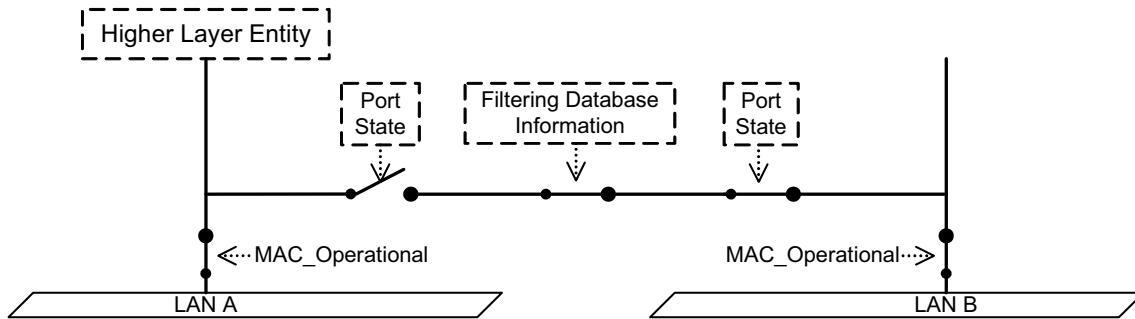


Figure 8-15—Single point of attachment—relay not permitted

In Figure 8-15, frames received from LAN A are received by the entity directly, but frames received from LAN B are not relayed by the Bridge and will only be received by the entity if another forwarding path is provided between LANs A and B. If the Discarding Port state shown resulted from spanning tree computation (and not from disabling the Administrative Bridge Port State), such a path will exist via one or more Bridges. If there is no active Spanning Tree path from B to A, the network has partitioned into two separate Bridged Local Area Networks, and the Higher Layer Entity shown is reachable only via LAN A.

Specific Higher Layer Entities can take notice of the Administrative Bridge Port State, as required by their specification. The Spanning Tree Protocol Entity is one such example—BPDUs are never transmitted or received on Ports with an Administrative Bridge Port State of Disabled.

If a Bridge Port's MAC Entity is not operational, a Higher Layer Entity directly attached at the Port will not be reachable, as Figure 8-16 illustrates. The Spanning Tree Protocol Entity ensures that the Port State is Discarding if the MAC_Operational (6.6.2) is FALSE even if the Administrative Bridge Port State is Enabled.

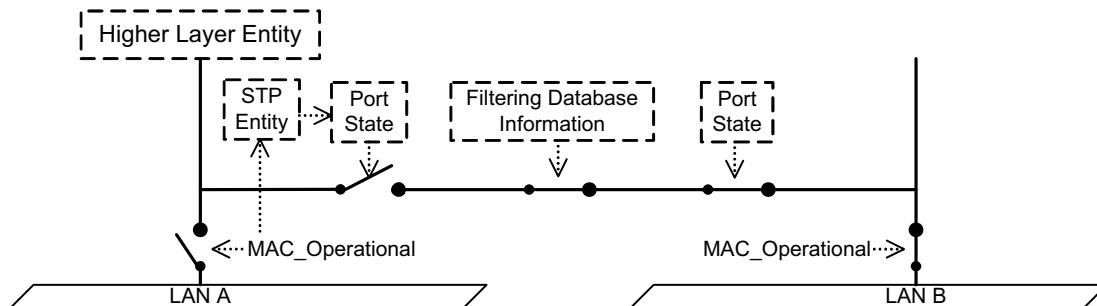


Figure 8-16—Effect of Port State

Port-based network access control (IEEE Std 802.1X) and MAC Security (IEEE Std 802.1AE [B7]) can be used to provide a further level of control over the connectivity provided by a Bridge Port to the MAC Relay Entity and the Higher Layer Entities within a bridge. Port-based network access control creates two distinct service access points for the LAN: the Controlled Port and the Uncontrolled Port. MAC_Operational is TRUE for the Controlled Port only when criteria for authentication, authorization, and secure connectivity have been satisfied; while MAC_Operational for the Uncontrolled Port is the same as that for direct access to the LAN. Both the Spanning Tree Protocol Entity and the MAC Relay Entity attach to the Controlled Port, thus ensuring that the connectivity seen by each of those entities is the same. If MAC_Operational for the Controlled Port is FALSE, the Spanning Tree Protocol Entity will ensure that both forwarding and learning (Clause 8.4) will be FALSE for that Port (i.e. the Port State will be Discarding). The Uncontrolled Port supports the operation of protocol entities such as the PAE specified by IEEE Std 802.1X, that

participate in the authentication exchanges necessary before Controlled Port connectivity is permitted or that distribute other unsecured information.

Figure 8-17 illustrates the connectivity provided to Higher Layer Entities if the MAC entity is physically capable of transmitting and receiving frames but MAC_Operational is FALSE for the Controlled Port. Higher Layer Entity A and the PAE are connected to an Uncontrolled Port and can transmit and receive frames using the MAC entity associated with the Port for LAN A, which Higher Layer Entity B cannot. None of the three entities can transmit or receive to or from LAN B.

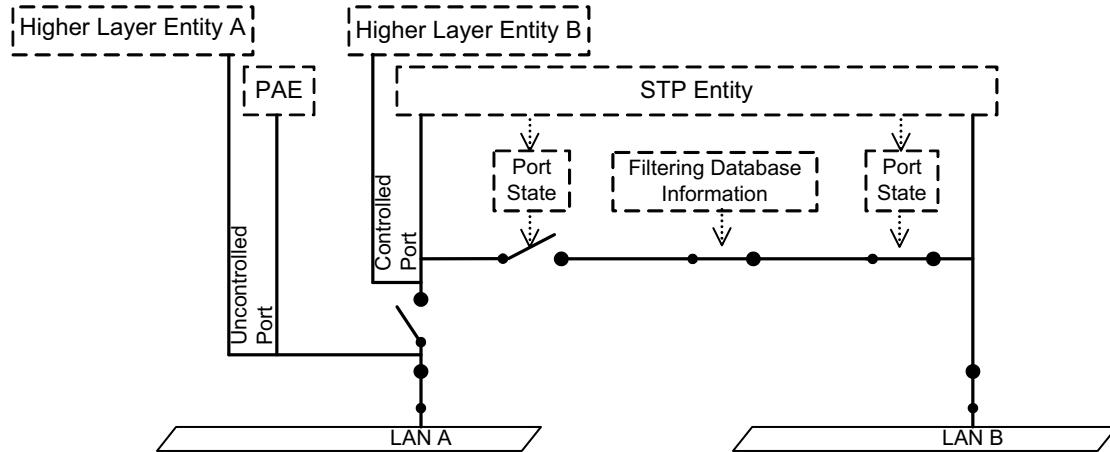


Figure 8-17—Controlled and Uncontrolled Port connectivity

NOTE 1—Reference should be made to IEEE Std 802.1X-2010 or later.

NOTE 2—The administrative and operational state values associated with the MAC, the Port's authorization state, and the Bridge Port State equate to the ifAdminStatus and ifOperStatus parameters associated with the corresponding interface definitions; see IETF RFC 2233 (1997) [B19].

8.13.10 VLAN attachment and connectivity for Higher Layer Entities

In VLAN-aware Bridges, two more switches appear in the forwarding path, corresponding to the actions taken by the Forwarding Process (8.6) in applying the ingress and egress rules (8.6.2 and 8.6.4), as illustrated in Figure 8-18.

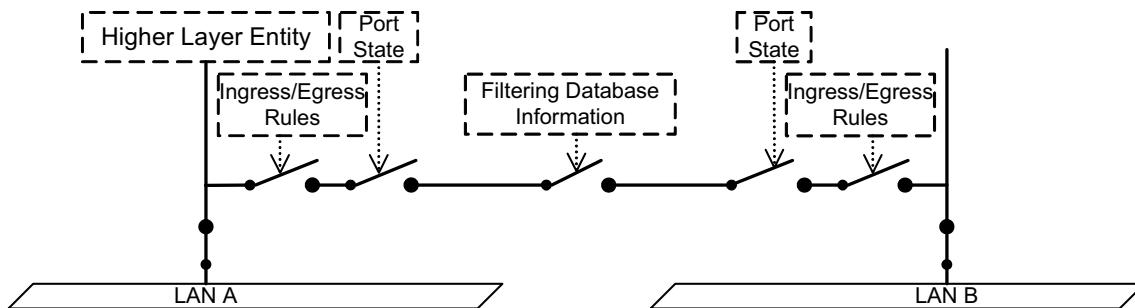


Figure 8-18—Ingress/egress control information in the forwarding path

As with Port state information, the configuration of the ingress and egress rules does not affect the reception of frames received on the same LAN as a Higher Layer Entity's point of attachment. For example, the reception of a frame by Higher Layer Entity A that was transmitted on LAN A is unaffected by the ingress or egress configuration of either Port. However, for Higher Layer Entities that require only a single point of

attachment, the ingress and egress configuration affects the forwarding path. For example, frames destined for Higher Layer Entity A that are transmitted on LAN B would be subjected to the ingress rules that apply to Port B and the egress rules that apply to Port A.

The decision as to whether frames transmitted by Higher Layer Entities are VLAN-tagged or untagged depends on the Higher Layer Entity concerned and the connectivity that it requires:

- a) Spanning Tree BPDUs transmitted by the Spanning Tree Protocol Entity are not forwarded by Bridges and must be visible to all other Spanning Tree Protocol Entities attached to the same LAN. Such frames shall be transmitted untagged;

NOTE—Any BPDUs or MVRPDUs that carry a tag header are not recognized as well-formed BPDUs or MVRPDUs and are not forwarded by the Bridge.

- b) The definition of the MVRP application (11.2.3) calls for all MVRP frames to be transmitted untagged for similar reasons;
- c) The definition of the MMRP application (Clause 10) calls for all MMRP frames originating from VLAN-aware devices to be transmitted VLAN-tagged, in order for the VID in the tag to be used to identify the VLAN context in which the registration applies;
- d) It may be necessary for PDUs transmitted for Bridge Management (8.12) to be VLAN-tagged in order to achieve the necessary connectivity for management in a Virtual Bridged Local Area Network. This is normally achieved by routing a packet containing the PDU to the routed subnet associated with the VLAN. Transmission of the packet through the router interface to that VLAN and subsequent forwarding of the resulting frame by VLAN-aware Bridges ensures that the frame is correctly VLAN-tagged, as required.

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 or more Individual MAC addresses, and one or more Group MAC addresses, in received frames, and to use one or more Individual MAC address or one or more 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). For VLAN based and backbone service instances, the Group MAC addresses are selected from Table 8-13 and Table 8-14 according to the MD Level and OpCode fields in the CFM PDU header. For these service instances, Loopback Messages (LBM, 20.2), Loopback Replies (LBR, 20.2), and Linktrace Replies (LTR, 20.3) are carried in unicast frames while Continuity Check Messages (CCM, 20.1) use addresses from Table 8-13, and Linktrace Messages (LTM, 20.3) use addresses from Table 8-14. In the case of TESI, all CFM messages use the Individual MAC Addresses or the Group MAC addresses that are associated with the monitored service (20.1, 20.2, 20.3).

NOTE—The Group destination MAC addresses in Table 8-13 are there primarily to make it possible for Bridges built prior to this standard to process CFM. This restriction on the choice of Group destination MAC addresses to be used on a CFM PDU is relaxed for ESP-VIDs. See 22.8.

Table 8-13—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 8-14—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

9. Tagged frame format

This clause specifies the format of the tags added to and removed from user data frames by the tag encoding and decoding functions that support the Enhanced Internal Sublayer Service (EISS, 6.8, 6.9, 6.10, 6.11) and the Backbone Service Instance Multiplex Entity (6.18). It

- a) Reviews the purpose of tagging, and the functionality provided;
- b) Specifies generic rules for the representation of tag fields and their encoding in the octets of a MAC Service Data Unit (MSDU);
- c) Specifies a general tag format, comprising a Tag Protocol Identifier (TPID), Tag Control Information (TCI), with additional information as signaled in the TCI, and;
- d) Specifies the format of the TPID for each IEEE 802 media access control method;
- e) Describes the types of tags that can be used, including the following:
 - 1) A Customer VLAN tag used at ports on a C-VLAN component,
 - 2) A Service VLAN tag used at ports on an S-VLAN component,
 - 3) A Backbone Service Instance tag used at Provider Instance Ports on an I-component and Customer Backbone Ports on a B-component;
- f) Documents the allocation of EtherType values to identify the types of tag specified in this standard;
- g) Specifies the format of the TCI and additional information for each tag type.

Further analysis of the frame formats and the format translations that can occur when frames are tagged or untagged when relayed between different media access control methods can be found in Annex G.

9.1 Purpose of tagging

Tagging a frame with a VLAN tag:

- a) Allows a VLAN Identifier (VID) to be conveyed, facilitating consistent VLAN classification of the frame throughout the network and enabling segregation of frames assigned to different VIDs;
- b) Allows priority (6.6, 6.8) to be conveyed with the frame when using IEEE 802 LAN media access control methods that provide no inherent capability to signal priority;

9.2 Representation and encoding of tag fields

In this subclause, octets are numbered starting from 1 and increasing in the order in which they are encoded in the sequence of octets that constitute a MAC Service Data Unit (MSDU).

Where bits in consecutive octets are used to encode a binary number in a single field, the lower octet number encodes the more significant bits of the field, and the LSB of the lower octet number and the most significant bit (MSB) of the next octet both form part of the field.

Where the value of a field comprising a sequence of octets is represented as a sequence of two-digit hexadecimal values separated by hyphens (e.g., A1-5B-03), the leftmost hexadecimal value (A1 in this example) appears in the lowest numbered octet of the field and the rightmost hexadecimal value (03 in this example) appears in the highest numbered octet of the field.

The bits in an octet are numbered from 1 to 8, where 1 is the LSB.

When the terms *set* and *reset* are used in the text to indicate the values of single-bit fields, *set* is encoded as a binary 1 and *reset* as a binary 0 (zero).

When the encoding of a field or a number of fields is represented using a diagram:

- a) Octets are shown with the lowest numbered octet nearest the top of the page, the octet numbering increasing from the top to bottom; or
- b) Octets are shown with the lowest numbered octet nearest the left of the page, the octet numbering increasing from left to right;
- c) Within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

9.3 Tag format

Each tag comprises the following sequential information elements:

- a) A Tag Protocol Identifier (TPID) (9.4);
- b) Tag Control Information (TCI) that is dependent on the tag type (9.5, 9.6);
- c) Additional information, if and as required by the tag type and TCI.

The tag encoding function supports each EISS (6.8) instance by using an instance of the Internal Sublayer Service (ISS) to transmit and receive frames and encodes the above information in the first and subsequent octets of the MSDU that will accompany an ISS M_UNITDATA.request, immediately prior to encoding the sequence of octets that constitute the corresponding EISS M_UNITDATA.request's MSDU. On reception the tag decoding function is selected by the TPID and decodes the TCI and additional information octets (if present) prior to issuing an EISS M_UNITDATA.indication with an MSDU that comprises the subsequent octets.

9.4 Tag Protocol Identifier (TPID) formats

The TPID includes an EtherType value that is used to identify the frame as a tagged frame and to select the correct tag decoding functions. Such an EtherType is known as the Tag EtherType.

Where the ISS instance used to transmit and receive tagged frames is provided by a media access control method that can support EtherType encoding directly (e.g., is an IEEE 802.3 MAC) or is media access method independent (e.g., 6.8), the TPID is EtherType encoded, i.e., is two octets in length and comprises solely the assigned EtherType value.

Where the ISS instance is provided by a media access method that cannot directly support EtherType encoding (e.g., is an IEEE 802.11 MAC), the TPID is encoded according to the rule for a Subnetwork Access Protocol (Clause 10 of IEEE Std 802) that encapsulates Ethernet frames over LLC, and comprises the SNAP header (AA-AA-03) followed by the SNAP PID (00-00-00) followed by the two octets of the assigned EtherType value.

9.5 Tag Protocol Identification

The following types of tags are specified:

- a) A customer VLAN tag (C-TAG), for general use by C-VLAN Bridges (5.9) and C-VLAN components of Provider Edge Bridges (5.10.1).
- b) a Service VLAN tag (S-TAG), reserved for use by S-VLAN Bridges (5.10), the S-VLAN component of Provider Edge Bridges (5.10.1) and Backbone Edge Bridges (5.11), and C-VLAN Bridges desiring to signal priority to a Provider Bridged Network or a Provider Backbone Bridged Network (6.13).
- c) A Backbone Service Instance tag (I-TAG), reserved for use by Backbone Edge Bridges (5.7, 5.8, 5.11).

NOTE 1—The Service VLAN tag (S-TAG) is identical to the Backbone VLAN tag (B-TAG, see 25.2).

A distinct EtherType has been allocated (Table 9-1) for use in the TPID field (9.4) of each tag type so they can be distinguished from each other, and from other protocols.

Table 9-1—IEEE 802.1Q EtherType allocations

Tag Type	Name	Value
Customer VLAN Tag	IEEE 802.1Q Tag Protocol EtherType (802.1QTagType)	81-00
Service VLAN Tag or Backbone VLAN Tag	IEEE 802.1Q Service Tag EtherType (802.1QSTagType)	88-a8
Backbone Service Instance Tag	IEEE 802.1Q Backbone Service Instance Tag EtherType (802.1QITagType)	88-e7

9.6 VLAN Tag Control Information

The VLAN TCI field (Figure 9-1) is two octets in length and encodes the `vlan_identifier`, `drop_eligible`, and priority parameters of the corresponding EISS M_UNITDATA.request as unsigned binary numbers.

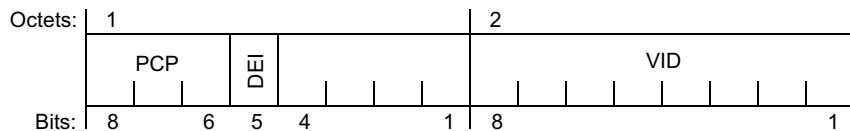


Figure 9-1—VLAN TCI format

The VID is encoded in a 12-bit field. A VLAN-aware Bridge may not support the full range of VID values but shall support the use of all VID values in the range 0 through a maximum N, less than or equal to 4094 and specified for that implementation. Table 9-2 identifies VID values that have specific meanings or uses.

Table 9-2—Reserved VID values

VID value (hexadecimal)	Meaning/Use
0	The null VID. Indicates that the tag header contains only priority information; no VID is present in the frame. This VID value shall not be configured as a PVID or a member of a VID Set, or configured in any Filtering Database entry, or used in any Management operation.
1	The default PVID value used for classifying frames on ingress through a Bridge Port. The PVID value of a Port can be changed by management.
2	The default SR_PVID value used for SRP (35.2.1.4(i)) Stream related traffic. The SR_PVID value of a Port can be changed by management.
FFF	Reserved for implementation use. This VID value shall not be configured as a PVID or a member of a VID Set, or transmitted in a tag header. This VID value may be used to indicate a wildcard match for the VID in management operations or Filtering Database entries.

NOTE—There is a distinction between the range of VIDs that an implementation can support and the maximum number of active VIDs supported at any one time. An implementation supports only 16 active VIDs, for example, may use VIDs chosen from anywhere in the identifier space, or from a limited range. The latter can result in difficulties where different

Bridges in the same network support different maximums. It is recommended that new implementations of this standard support the full range of VIDs, even if the number of active VIDs is limited.

The priority and drop_eligible parameters are conveyed in the 3-bit Priority Code Point (PCP) field and the Drop Eligible Indicator (DEI) field as specified in 6.9.3.

NOTE—Previous versions of this standard used bit 5 of octet 1 of the TCI in Customer VLAN tags as a Canonical Format Indicator (CFI). Bridges conformant to this version of the standard will not interoperate with bridges that set the CFI as a result of inserting Customer VLAN tags in frames received from an 802.5 Token Ring LAN. Bridges conformant to this version of the standard will interoperate with bridges that forward bit 5 of octet 1 as a CFI but do not have 802.5 Token Ring interfaces.

9.7 Backbone Service Instance Tag Control Information

The I-TAG TCI field (Figure 9-3) is 16 octets in length and encodes the priority, drop_eligible, destination_address, and source_address parameters of the corresponding service request primitive as unsigned binary numbers. Backbone Service Instance tags are encoded and decoded at Provider Instance Ports and Customer Backbone Ports of Backbone Edge Bridges as specified in 6.10, 6.11, and 6.18.

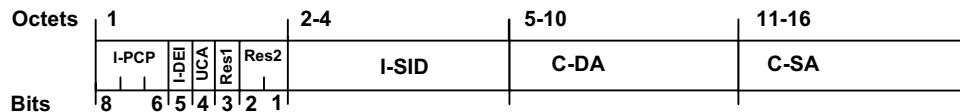


Figure 9-3—I-TAG TCI format

The I-TAG TCI contains the following fields:

- a) *Priority Code Point (I-PCP)*—This 3-bit field encodes the priority and drop_eligible parameters of the service request primitive associated with this frame using the same encoding as specified for VLAN tags in 6.9.3. The Provider Backbone Bridged Network operates on the priority associated with the B-TAG.
- b) *Drop Eligible Indicator (I-DEI)*—This 1-bit field carries the drop_eligible parameter of the service request primitive associated with this frame. The Provider Backbone Bridged Network operates on the drop eligibility associated with the B-TAG.
- c) *Use Customer Addresses (UCA)*—A single-bit flag that, when containing a value of one, signals a Backbone Service Instance Multiplex Entity (6.18) to use the addresses contained in the C-DA and C-SA fields.
- d) *Reserved 1 (Res1)*—This 1-bit field is used for any future format variations. The Res1 field contains a value of zero when the tag is encoded, and is ignored when the tag is decoded.
- e) *Reserved 2 (Res2)*—This 2-bit field is used for any future format variations. The Res2 field contains a value of zero when the tag is encoded. The frame will be discarded if this field contains a nonzero value when the tag is decoded.
- f) *Backbone Service Instance Identifier (I-SID)*—This 24-bit field carries the Backbone Service Instance Identifier of the backbone service instance.
- g) *Encapsulated Customer Destination Address (C-DA)*—Contains the address in the destination_address parameter of the service request primitive associated with this frame. The address is represented using the Hexadecimal Representation as specified in IEEE Std 802 with the octet containing the I/G bit in the lowest numbered octet of the field.
- h) *Encapsulated Customer Source Address (C-SA)*—Contains the address in the source_address parameter of the service request primitive associated with this frame. The address is represented

using the Hexadecimal Representation as specified in IEEE Std 802 with the octet containing the I/G bit in the lowest numbered octet of the field.

The Backbone Service Instance Identifier is encoded in a 24-bit field. Table 9-3 identifies I-SID values that have specific meanings or uses.

Table 9-3—Reserved I-SID values

I-SID value (hexadecimal)	Meaning/Use
0	Reserved for use by future amendments to this standard.
1	Default value—Unassigned ISID on a VIP.
2 through FF	Reserved for use by future amendments to this standard.
FFFFFF	Reserved for implementation use. This I-SID value shall not be configured as an identifier for a backbone service instance or transmitted in a Backbone Service Instance Tag header. This I-SID value may be used to indicate a wildcard match for the I-SID in management operations.

NOTE—There is a distinction between the range of I-SIDs that an implementation can support, and the maximum number of active backbone service instances supported at any one time. An implementation that supports only 2^{16} active backbone service instances, for example, may use I-SIDs chosen from anywhere in the identifier space. Implementations of this standard support the full range of I-SIDs, even if the number of active backbone service instances is limited.

10. Multiple Registration Protocol (MRP) and Multiple MAC Registration Protocol (MMRP)

The Multiple Registration Protocol allows participants in an MRP application to register attributes with other participants in a Bridged Local Area Network. The definition of attribute types, their values, and the semantics associated with values when registered, are specific to each MRP application.

This clause

- a) Provides an overview of the use of MRP within a bridged network (10.1)
- b) Describes the architecture of MRP participants for end stations and Bridge Ports (10.2)
- c) Specifies registration propagation between the per-Port Participants in a Bridge (10.3)
- d) Details requirements to be met by the MRP design (10.4)
- e) Details requirements for interoperability between MRP participants (10.5)
- f) Provides an overview of protocol operation (10.6)
- g) Provides a detailed specification of the protocol (10.7)
- h) Describes the structure of protocol data units exchanged between MRP participants (10.8)

Subclauses 10.9 through 10.12 define an MRP application, the Multiple MAC Registration Protocol (MMRP), that registers attributes of two types—MAC Addresses and Group service requirements. Values of these attributes control MAC address filtering by MMRP participants.

Clause 11 defines a second MRP application, the Multiple VLAN Registration Protocol (MVRP), that registers VLAN membership information.

Clause 35 defines a third MRP application, the Multiple Stream Reservation Protocol (MSRP), that registers data Stream characteristics and reserves Bridge resources as appropriate to provide QoS guarantees.

10.1 MRP overview

MRP allows a participant in a given MRP application to make or withdraw *declarations of attributes*, and for those declarations (or withdrawals) to result in the *registration* (or removal of registrations) of those attributes with the other MRP Participants for that application.

A declaration by an MRP Participant for an end station or Bridge Port is recorded by an Applicant state machine for the declared attribute and Port. Changes in the Applicant state machine's variables trigger the transmission of MRPDUs to communicate the declaration (or withdrawal).

A registration is recorded by a Registrar state machine for the attribute at each participating end station and Bridge Port that receives the MRPDU. Removal of a given attribute registration occurs only if all the other participants connected to the same LAN withdraw the declaration.

Attributes registered on Bridge Ports that are part of the applicable *active topology* (8.4, 10.3.1) are declared on all the other Bridge Ports that are also part of that active topology. Hence, a given declaration is propagated to all application participants, and registered in each Bridge on those Ports that are “nearest” to the source or sources of the declaration within the active topology.

Figure 10-1 illustrates the result of a single end station making a declaration, and shows the Bridge Ports that also make declarations to propagate the attribute. The attribute is propagated to all LANs in the Bridged Local Area Network, but the directional nature of the propagation results in registration only on Bridge Ports that receive (as opposed to transmit) declarations.

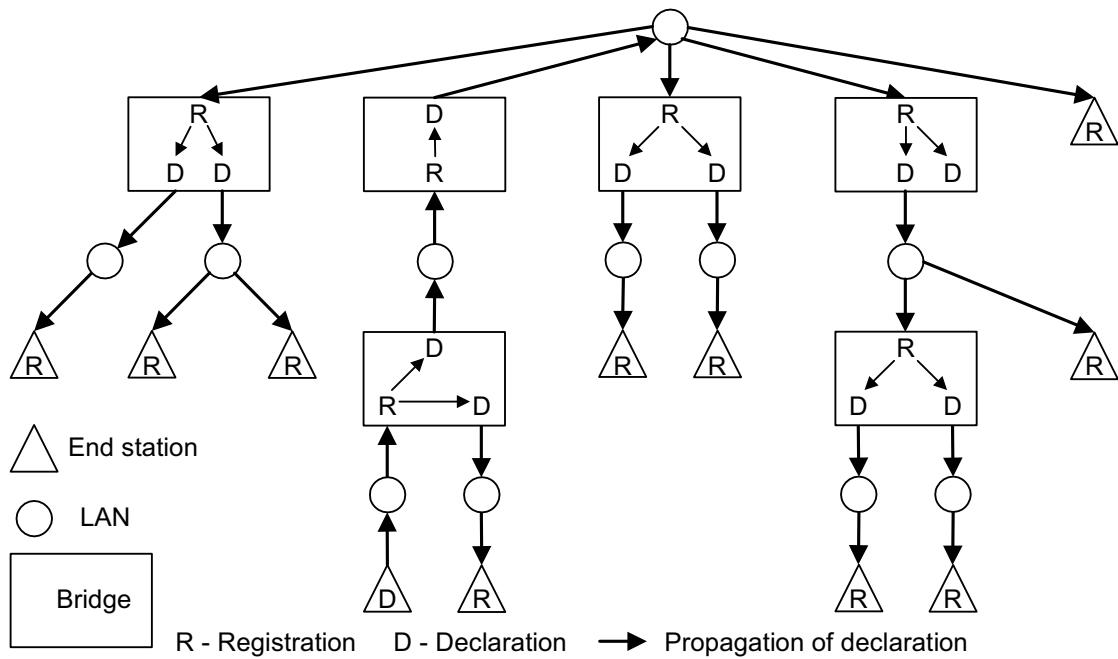


Figure 10-1—Example—Attribute value propagation from one station

NOTE—Unless otherwise stated, the following description assumes operation within the Base Spanning Tree Context (see 10.3.1). While registration can occur on any Bridge Port, regardless of Port State (8.4), propagation follows the spanning tree active topology. All the Bridge Ports shown in Figure 10-1, Figure 10-2, and Figure 10-3 are in the Forwarding Port State.

Figure 10-2 illustrates the result of different end stations declaring the same attribute on different LANs. All end stations register the attribute, and some Bridges register it on more than one Port.

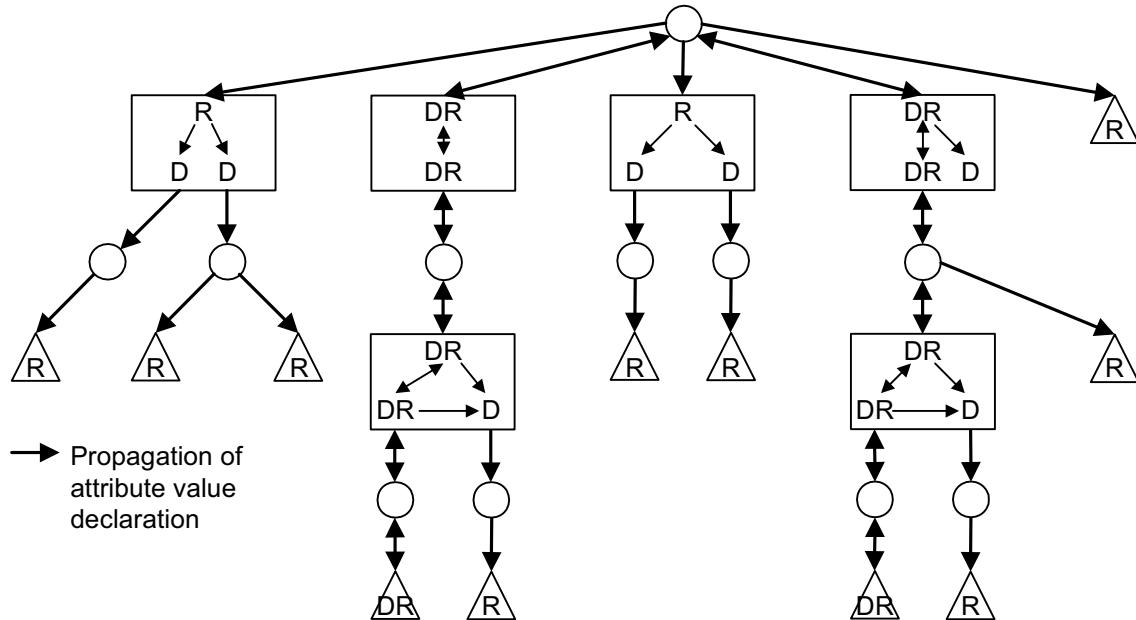


Figure 10-2—Example—Attribute value propagation from two stations

The set of Bridge Ports and end stations that both declare and register a given attribute defines the subset of the active topology that contains all the participants declaring that attribute. A registration can be regarded as a pointer to participants that have declared that attribute, as illustrated in Figure 10-3 (using the same set of declarations and registrations that were illustrated in Figure 10-2).

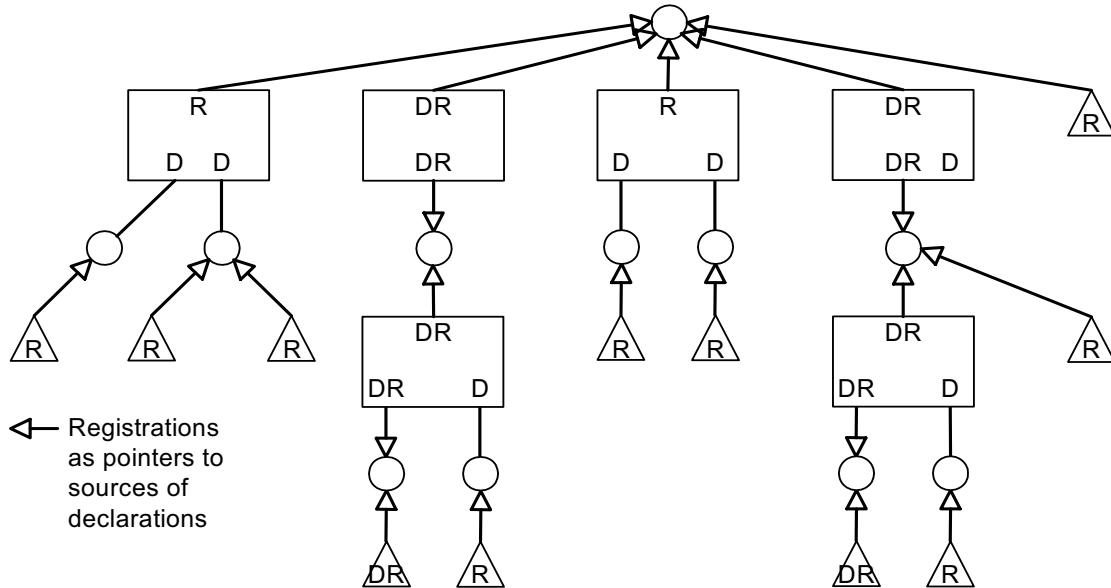


Figure 10-3—Example—Registrations as pointers to the sources of declarations

Situations where it is desirable to form “reachability trees” are generally good candidates for the use of MRP. For example, if the attribute in Figure 10-3 is a Group MAC Address that is used as a destination address for a video data stream, and it is deemed desirable for that video data to be sent only to the subset of the active topology that contains end stations that have declared that attribute, then an end station that is the source of that video stream could use the presence or absence of a registration as an indication of whether or not to send the data on the LAN to which it is attached. Any Bridge receiving the data could determine on which Ports the data should be forwarded.

VLAN Bridges that do not support MRP are transparent to MRP exchanges, and forward received MRPDUs on all Ports that are in Forwarding. Similarly, VLAN Bridges that do not implement a given MRP application are transparent to MRP exchanges destined for that application, and forward any such received MRPDUs on all Ports that are in Forwarding.

MRP operates only on Ports that are MAC_Operational (6.6.2). If the Port is operating as a network access port (IEEE Std 802.1X), MRP uses the controlled port (8.13.9). On any Port whose MAC_Operational parameter is FALSE, any MRP entity shall not transmit MRPDUs, and shall discard, without processing, any received MRPDUs.

MRP provides a means to mark initial attribute declarations and propagated attribute declarations as “new,” signaling to the recipient of the declaration that the attribute value is being newly declared, or is being redeclared following a change in the underlying topology. The rules applied to the marking and propagation of newly declared values in this way are common to all MRP Applications; however, the action taken on receipt of an attribute declaration marked as “new” is specific to each MRP Application. For example, MMRP (10.9) makes no use of the “new” marking, whereas MVRP (Clause 11) uses the “new” marking to permit Filtering Database entries to be flushed on a per-VID basis following a topology change.

10.2 MRP architecture

An MRP Participant consists of an application component, and an MRP Attribute Declaration (MAD) component. The application component is responsible for the semantics associated with Attribute values and their registration, including the use of explicitly signaled new declarations, and uses the following two primitives to request MAD to make or withdraw Attribute declarations:

MAD_Join.request (attribute_type, attribute_value, new)

MAD_Leave.request (attribute_type, attribute_value)

where *attribute_type* specifies the type of the attribute declaration, and *attribute_value* specifies the instance of that type, and the Boolean *new* parameter indicates an explicit new declaration. If the value of tcDetected (13.23) for the Port and MAP Context associated with the MRP Participant is nonzero, then the value of the *new* parameter in the propagated MAD_Join.request is set TRUE.

The MAD component executes MRP (10.6, 10.7), generating MRP messages for transmission and processing messages received from other Participants, and uses the following two primitives to notify its application component of a change in Attribute registration:

MAD_Join.indication (attribute_type, attribute_value, new)

MAD_Leave.indication (attribute_type, attribute_value)

One such Participant, per MRP application, exists for each point of attachment to a LAN where Attributes for that application are to be declared or registered, i.e., one Participant per application in an end station, and one per application per Port in a Bridge. The encoding of Attribute values in MRPDUs and their subsequent decoding is application specific, within the general structure specified in 10.8, and each MRPDU conveys messages generated by the MAD component for a single application.

Within a Bridge, a per application MRP Attribute Propagation (MAP) component (10.3) propagates information between the per-Port Participants, using the same request and indication primitives.

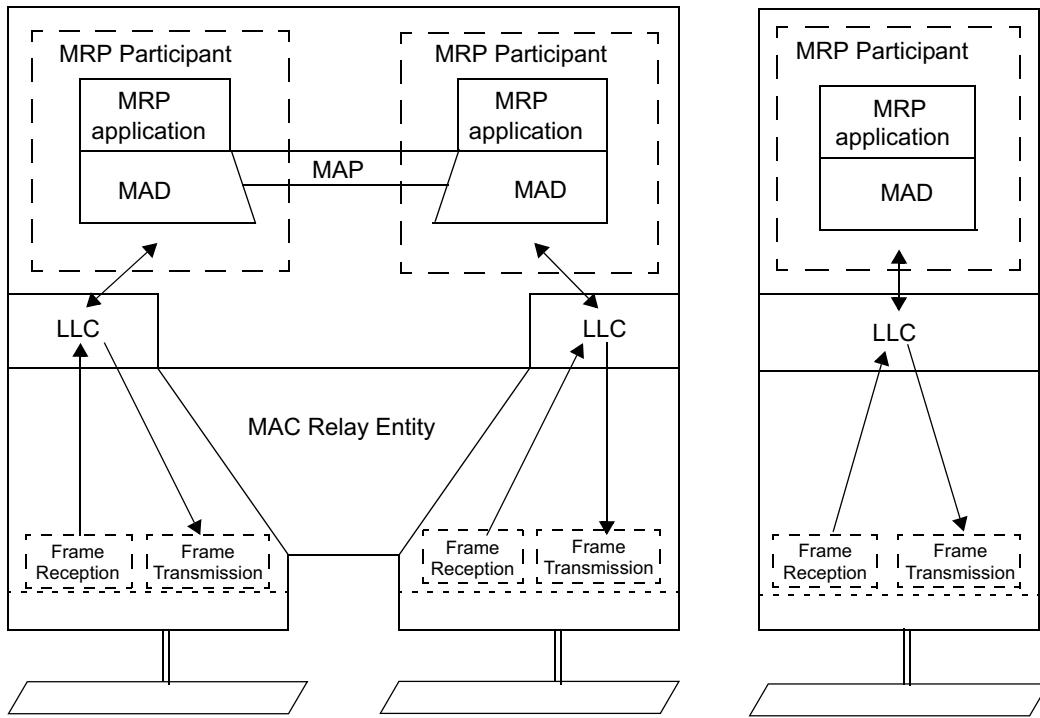
Figure 10-4 illustrates the components of MRP Participants in a two-Port Bridge and an end station.

For each MRP application, the following are defined:

- a) A set of Attribute types used by the application.
- b) The Attribute values permitted for each Attribute type.
- c) The semantics associated with each Attribute type and value.
- d) The use made of MAP Contexts by the application.
- e) The group MAC address and EtherType for protocol exchanges between application Participants.
- f) The structure and encoding of the Attribute types and values in MRPDUs.
- g) The requirements for MRP state machine support in end stations and Bridges.
- h) The circumstances, if any, in which the application makes use of the “new” declaration capability.

NOTE 1—Not all applications of MRP will make use of the “new” declaration capability.

- i) If the application makes use of the “new” declaration capability, the semantics that new registrations carry in that application.
- j) The number of attribute values out of the possible range of values for which the application is expected to be capable of maintaining current state information.

**Figure 10-4—MRP architecture**

NOTE 2—In some applications, such as MVRP, the failure of one bridge in a network to maintain state for all possible attribute values would have major consequences for the integrity of the network. For such applications, the application definition will mandate the ability to track the state of all possible attribute values.

10.3 MRP Attribute Propagation (MAP)

The MRP Attribute Propagation (MAP) function enables propagation of attributes registered on Bridge Ports across the network to other participants. Each MRP application specifies the operation of the MAP function. This subclause specifies the operation of the MAP function for the MMRP application and the MVRP application (11.2.1). The MAP function for the MSRP application is specified in 35.2.4.

For a given MRP application and MAP Context (10.3.1), and for the set of Ports that are in a Forwarding state as defined by that MAP Context:

- Any MAD_Join.indication, or any MAD_Join.request issued by the MRP application, received by MAP from a given Port in the set is propagated as a MAD_Join.request to the instance(s) of MAD associated with each other Port in the set. If the value of tcDetected (13.23) for the Port and MAP Context is nonzero, then the value of the *new* parameter in the propagated MAD_Join.request is set TRUE, regardless of the value of this parameter in the indication or request that is being propagated.
- Any MAD_Leave.indication, or any MAD_Leave.request issued by the MRP application, received by MAP from a given Port in the set is propagated as a MAD_Leave.request to the instance(s) of MAD associated with each other Port in the set (Port P, say) if and only if no registration now exists for that Attribute on any other Port in the set excluding P.

These rules propagate attribute registrations through any given Port if any other Port has seen a registration for that attribute, and propagate de-registrations if all other Ports are now deregistered.

As the set of Ports that are in a Forwarding state for a given MAP Context can change dynamically, for example as a result of Spanning Tree reconfiguration, MAP operates as follows after such a change:

- c) If a Port is added to the set, and that Port has registered an attribute (i.e., a MAD_Join.indication or MAD_Join.request has occurred more recently than any MAD_Leave.indication or MAD_Leave.request for the attribute), then MAD_Join.requests are propagated to the MAD instances for each of the other Ports in the set.
- d) If a Port is added to the set, but that Port has not declared an attribute that other Ports in the set have registered, then MAD_Join.requests are propagated by those other Ports to the MAD instance for that Port.
- e) If a Port is removed from the set, and that Port has registered an attribute and no other Port has, then MAD_Leave.requests are propagated to the MAD instances for each of the other Ports in the set.

NOTE—If a Port is removed from the set, and that Port has declared one or more attributes, then this Port transmits a Leave message (see 10.6) for every attribute that it has declared.

- f) If a Port is removed from the set, and that Port has registered an attribute that another Port has also registered, then a MAD_Leave.request is propagated to the MAD instance for that other Port.

10.3.1 MAP Context

For a given Port of an MRP-aware Bridge and MRP application supported by that Bridge, an instance of an MRP Participant can exist for each *MRP Attribute Propagation Context (MAP Context)* understood by the Bridge. A MAP Context identifies the set of Bridge Ports that form the applicable *active topology* (8.4).

Examples of MAP Contexts are as follows:

- a) The active topology formed by the operation of RSTP (Clause 13). This MAP Context provides the same connectivity as the Spanning Tree in each Bridge and is known as the *Base Spanning Tree Context*.
- b) The active topology formed by the subset of Bridge Ports, and the underlying Spanning Tree, that support a given VID. This MAP Context is known as a *VLAN Context*.

NOTE—This standard uses the Base Spanning Tree Context and VLAN Contexts to define the operation of MMRP and MVRP; however, other MAP Contexts may be used for other applications, or for extending MMRP functionality.

MRP exchanges can occur on all of the Ports of a Bridge; however propagation across a Bridged Local Area Network of attribute registrations for a given application uses only those Bridge Ports that are part of the active topology identified by the MAP Context. Each MRP application specification identifies the contexts it can operate within, specifies rules for selecting forwarding Ports, and assigns MAP Context identifiers for use in conjunction with the operation of MRP and its administrative controls (10.7.2, 10.7.3).

A MAP Context identifier of 0 always identifies the *Base Spanning Tree Context*. The MRP application specifies how the MAP Context of each MRPDU is identified if any other context is used.

10.3.1.1 MAD and Port role changes

A spanning tree provides a loop-free active topology connecting all the bridges and all the LANs in a network. Hence, any Bridge Port that is in a forwarding state connects two otherwise disconnected parts of the topology. Each Alternate Port on a Bridge A (say) connects to a LAN that has a Port of another Bridge with a better priority vector than A can supply; hence, each of A's Alternate Ports is a potential connection to part of the network that is not connected to any of A's Designated Ports, all LANs and Bridges in any such part having an equal or worse priority vector than A. Therefore, an Alternate Port on A potentially connects to the part of the network connected to by A's Root Port. Since any part of the network is fully connected, in

the absence of registration controls, A's Alternate Port will have registered the same attributes as its Root Port.

For implementations running over RSTP or MSTP, this gives rise to the risk of information loops when Port roles change; because of the store and forward nature of attribute propagation and the potentially rapid transitions of Port roles (compared to the relatively slow transitions that occurred with STP), these can arise even when there are no data loops.

To prevent such information loops from occurring, the information held by MAD's Registrars for a Port (i.e., information registered on a Port as a result of protocol activity on the LAN to which that Port is connected) is discarded whenever the Port transitions from an Alternate port or Root Port role to become a Designated Port. No such discard is needed for changes in the other direction, i.e., changes from Designated Port to Root Port or Alternate Port.

10.4 Requirements to be met by MRP

MRP establishes, maintains, withdraws, and disseminates attribute declarations and registrations among the MRP Participants attached to a single LAN. The protocol meets the following requirements for Applicant and Registrar behavior, error recovery, performance, scalability, compatibility with non-MRP aware devices, and the load imposed on Bridges, end stations, and the network:

- a) Participants can issue declarations for MRP application attributes (10.2, 10.3, 10.7.3, and 10.7.7).
- b) Participants can withdraw declarations for attributes (10.2, 10.3, 10.7.3, and 10.7.7).
- c) Each Bridge propagates declarations to MRP Participants (10.3).
- d) MRP Participants can track the current state of declaration and registration of attributes on each Port of the participant device (10.7.7 and 10.7.8).
- e) MRP Participants can remove state information relating to attributes that are no longer active within part or all of the network, e.g., as a result of the failure of a participant (10.7.8 and 10.7.9).
- f) The latency involved in issuing, propagating, or revoking attribute declarations, is small (i.e., comparable to the frame propagation delay) and increases linearly as a function of the diameter of the network (10.7.7, 10.7.8, and 10.7.9).
- g) MRP is resilient in the face of the failure of MRP Participants.
- h) MRP is resilient in the face of single packet loss.
- i) MRP will operate correctly in networks where
 - 1) All Bridges support both Basic and Extended Filtering Services; or where
 - 2) Some Bridges support only Basic Filtering Services and some both Basic and Extended Filtering Services (10.5, 10.7.7, 10.7.8, and 10.7.9).
- j) The communications bandwidth consumed on any particular LAN by Applicants and Registrars in exchanging MRPDUs will be a small percentage of the total available bandwidth, and independent of the total traffic supported by the network. The bandwidth consumed will be a function of the number of attributes registered.

10.5 Requirements for interoperability between MRP Participants

To ensure the interoperability of MRP, the following are required:

- a) All MRP applications use a group MAC address as the destination address of MRPDUs, selected in accordance with the stated requirements of the MRP application concerned for the type of bridge component in which the application is implemented. The addresses used may be taken from the set of addresses specified in Table 10-1, or taken from the set of reserved addresses specified in Table 8-1, Table 8-2, and Table 8-3, or other group MAC addresses, chosen according to the particular properties required by the application concerned.

NOTE 1—The addresses in Table 8-1, Table 8-2, Table 8-3, and Table 10-1 differ from other group MAC addresses in terms of the scope of transmission within a network, as a consequence of the forwarding/filtering decisions that are taken relative to them by different types of bridge component.

- b) Table 10-1 specifies a set of group MAC addresses, some that have been assigned for use by existing applications defined in this standard, and the others reserved for future standards use. Addresses in this set have the property that
 - 1) Where a given address in the set is used by a bridge component to support an MRP application, frames destined for that address shall not be forwarded by that bridge component; i.e., a static filtering entry for that address is maintained for that group MAC address in order to prevent the forwarding of frames destined for that address.
 - 2) Where a given address in the set is not used by a bridge component to support any MRP application, frames destined for that address received on any Port that is part of a given active topology shall be forwarded by that bridge component on all other Ports that are part of that active topology.
- c) The transmission and reception of MRPDUs between MRP Participants, formatted as defined for the application using the generic PDU format defined in 10.8, shall use LLC procedures. Each MRP application uses a unique EtherType value in order to identify the application protocol. Table 10-2 specifies the EtherType values assigned to existing applications.

NOTE 2—For the purposes of this standard, the expression “use LLC procedures” includes making use of the service provided by link layer protocol entities that support protocol discrimination by means of an EtherType value.

- d) MRPDUs, i.e., frames with the destination MAC addresses selected as specified in item a) and the EtherType values specified in item c), that are destined for applications supported by a bridge component, and that are not well formed (i.e., are not structured and encoded as defined in 10.8 and with attribute types and values encoded as defined by the MRP application), shall be discarded on receipt.

Table 10-1—MRP application addresses

Assignment	Value
Customer and Provider Bridge MMRP address	01-80-C2-00-00-20
Customer Bridge MVRP address	01-80-C2-00-00-21
Reserved	01-80-C2-00-00-22
Reserved	01-80-C2-00-00-23
Reserved	01-80-C2-00-00-24
Reserved	01-80-C2-00-00-25
Reserved	01-80-C2-00-00-26
Reserved	01-80-C2-00-00-27
Reserved	01-80-C2-00-00-28
Reserved	01-80-C2-00-00-29
Reserved	01-80-C2-00-00-2A
Reserved	01-80-C2-00-00-2B
Reserved	01-80-C2-00-00-2C
Reserved	01-80-C2-00-00-2D
Reserved	01-80-C2-00-00-2E
Reserved	01-80-C2-00-00-2F

Table 10-2—MRP EtherType values

Assignment	Value
MMRP EtherType	88-F6
MVRP EtherType	88-F5
MSRP EtherType	22-EA

10.6 Protocol operation

This subclause provides an informal introduction. The definitive specification of MRP is contained in 10.7 and 10.8.

MRP is a simple, fully distributed, many-to-many protocol, that supports efficient, reliable, and rapid declaration and registration of attributes by multiple participants on shared and virtual shared media. MRP also incorporates optimizations to speed attribute declarations and withdrawals on point-to-point media. Correctness of MRP operation is independent of the relative values of protocol timers, and the protocol design is based primarily on the exchange of idempotent protocol state rather than commands.

A full MRP Participant maintains Registrar and Applicant state machines for each Attribute of interest, and a LeaveAll state machine and PeriodicTransmission state machine for the participant as a whole.

The job of the Registrar is to record declarations of the attribute made by other Participants on the LAN. It does not send any protocol messages, as the Applicant looks after the interests of all would-be Participants.

The job of the Applicant is twofold:

- a) To ensure that this Participant's declaration is correctly registered by other Participants' Registrars.
- b) To prompt other Participants to reregister after one withdraws a declaration.

NOTE 1—Applicant-Only implementations are concerned only with item a).

The basic design of MRP is oriented to LAN environments, where there is generally a low probability of frame loss, and ensures that timely registration of attributes is unaffected unless at least two out of a set of immediately related frames are lost. The LeaveAll state machine periodically ensures that Participants reregister attributes, thus guarding against an extended failure to register or deregister. In potentially lossy environments such as service instances supported by Provider Bridged Networks, where frame losses are temporally correlated, the PeriodicTransmission machine ensures successful registration without immediately contributing to congestive loss.

If one Applicant has both declared and registered an Attribute, other Applicants do not need to reiterate the declaration. MRP messages convey both Applicant and Registrar state to allow suppression of such repeated declarations. The Applicant state machine distinguishes between Active Participants (that have sent a message or messages to make a declaration), Passive Participants (that require registration, but have not had to declare the attribute so far to register it, and will not have to explicitly deregister), and Observers (that do not require registration at present, but track the attribute's registration in case they do and become Passive Participants). The following four distinct messages communicate the transmitting participant's state for an Attribute:

- Empty—Not declared, and not registered.
- In—Not declared, but registered.
- JoinEmpty—Declared, but not registered.
- JoinIn—Declared and registered.

One message communicates a withdrawal of a prior declaration, so that Registrars do not have to track individual Applicants:

- Leave—Previously registered, but now withdrawn.

A further message conveys a declaration from a new Participant, or a Participant newly added to the active topology reached through a Bridge Port. This information is propagated by the MAP component, and so includes additional registrations elsewhere in the bridged network:

- New—Newly declared, and possibly not previously registered.

The LeaveAll state machine uses the following single message that applies to all Attributes.

- LeaveAll—All registrations will shortly be deregistered; Participants need to reregister.

The Registrar for each Attribute actually implements the following three states for the Attribute:

- IN—Registered.
- LV—Previously registered, but now being timed out.
- MT—Not registered.

A single timer, the leavetimer, is associated with each Registrar and operates in the LV state. In that state, MRP messages for the Attribute report it as unregistered (using Empty or Join Empty) but the MAD Leave indication is delayed until the leavetimer expires and the state transitions to MT.

NOTE 2—The accuracy required for the leavetimer is sufficiently coarse as to permit the use of a single operating system timer per Participant with 2 bits of state for each Registrar.

The Applicant for each Attribute implements states that record whether it wishes to make a new declaration, to maintain or withdraw an existing declaration, or has no declaration to make. It also records whether it has actively made a declaration, or has been passive, taking advantage of or simply observing the declarations of others. It counts the New, JoinIn, and JoinEmpty messages it has sent, and JoinIn messages sent by others, to ensure that at least two such messages have been sent since it last received a LeaveAll or Leave message, and at least one since it last received a JoinEmpty or Empty message. This ensures that each of the other Participant's Registrars for the Attribute either have received (assuming no packet loss) two Join or New messages or have reported the Attribute as registered. The Applicant state machine (Table 10-3) uses the following states:

- VO—Very anxious Observer. The applicant is not declaring the attribute, and has not received a JoinIn message since the state machine was initialized, or since last receiving a Leave or LeaveAll.
- VP—Very anxious Passive. The applicant is declaring the attribute, but has neither sent a Join nor received a JoinIn since the state machine was initialized, or since last receiving a LeaveAll or Leave.
- VN—Very anxious New. The applicant is declaring the attribute, but has not sent a message since receiving a MAD Join request for a new declaration.
- AN—Anxious New. The applicant is declaring the attribute, and has sent a single New message since receiving the MAD Join request for the new declaration.
- AA—Anxious Active. The applicant is declaring the attribute, and has sent a Join message, since the last Leave or LeaveAll, but either has not received another JoinIn or In, or has received a subsequent message specifying an Empty registrar state.
- QA—Quiet Active. The applicant is declaring the attribute and has sent at least one of the required Join or New messages since the last Leave or LeaveAll, has seen or sent the other, and has received no subsequent messages specifying an Empty registrar state.

- LA—Leaving Active. The applicant has sent a Join or New message since last receipt of a Leave or LeaveAll, but has subsequently received a MAD Leave request and has not yet sent a Leave message.
- AO—Anxious Observer. The applicant is not declaring the attribute, but has received a JoinIn since last receiving a Leave or LeaveAll.
- QO—Quiet Observer. The applicant is not declaring the attribute, but has received two JoinIns since last receiving a Leave or LeaveAll, and at least one since last receiving a message specifying an Empty registrar state.
- AP—Anxious Passive. The applicant is declaring the attribute, and has not sent a Join or a New since last receiving a Leave or a LeaveAll but has received messages as for the Anxious Observer state.
- QP—Quiet Passive. The applicant is declaring the attribute, and has not sent a Join or a New since last receiving a Leave or a LeaveAll but has received messages as for the Quiet Observer state.
- LO—Leaving Observer. The applicant is not declaring the attribute, and has received a Leave or LeaveAll message.

If an Applicant receives a Leave or LeaveAll message it will send a JoinEmpty or Empty message, and that message will prompt any other Applicants that are declaring the attribute to send a JoinIn. An Applicant that sends a LeaveAll message will also ensure that an Empty message is sent to prompt other Applicants to repeat their declaration. Thus, if any Participant's Registrar deregisters an Attribute, at least two messages will be lost before another Participant's Applicant fails to make a required redeclaration.

To facilitate and encourage the transmission of timely protocol information and the encoding of messages for multiple Attributes within the same MRPDU, rather than the use and subsequent queuing of multiple PDUs prior to transmission, PDU transmission is specified in terms of requests for a transmission opportunity, and the Applicant and LeaveAll state machines specify the necessary addition of messages to an MRPDU when that opportunity occurs. Whenever a state machine transitions to a state that requires transmission of a message, a transmit opportunity is requested if one is not already pending. The message actually transmitted (if any) is that appropriate to the state of the machine when the opportunity is presented.

NOTE 3—Specifying transmit opportunity requests and their subsequent use is also intended to aid correct implementation in systems where a transmission is not possible immediately, e.g., shortly after whole or part of the system is initialized when other claims on system resources take precedence or interfaces are not yet available.

To support the efficient encoding of many messages in a single MRPDU, the number of message types has been kept to the minimum consistent with protocol goals and requirements, and the state machines specify both whether it is necessary to send a message and a message type that can always be sent—thus removing the need for a further “no message” encoding. The MRPDU structure and compact encoding (10.8) allows all potential attributes for certain MRP applications, e.g., MVRP (Clause 11), to be encoded in a single IEEE 802.3 frame and this further promotes protocol efficiency through ready detection of missing registrations. Protocol efficiency and correct registration are further supported by encoding a LeaveAll message (if required) at the beginning of the MRPDU, and by using an Applicant state machine that minimizes state changes for those Attributes whose redeclaration can be encoded in the same PDU—if the LeaveAll is received by another Participant so also will be the redeclaration, and sequential processing of the encoded messages by the recipient will ensure that the registration is uninterrupted. Applicant states for Attributes that cannot be redeclared in the same PDU allow for receipt of the LeaveAll but loss of the redeclaration.

LeaveAll messages are transmitted to ensure that any failure to withdraw a declaration does not result in an unwanted permanent registration—perhaps the system or process responsible for the registration has been removed from the LAN or has ceased to operate. Attributes registered by the Participant transmitting the LeaveAll as well as those receiving it are therefore timed out. The LeaveAll state machine (10.7.9) for the Participant operates a single timer, the leaveAllTimer, that causes a transmit opportunity to be requested when it expires, and transmission of a LeaveAll at the next opportunity. Reception of a LeaveAll message

from another Participant causes the timer to be restarted without generating a message, thus suppressing multiple LeaveAll messages from Participants connected to the same LAN.

NOTE 4—In the face of changing inputs, arbitrary loss, delay, reordering, and unsignaled interruption in participation, no protocol will meet its objectives. What can be said is that the objectives will be met after a known period of operation within specified limits, and what failures are most likely otherwise. MRP favors ensuring that registrations are made, at the expense of tolerating prolonged registrations. Thus, the LeaveAll mechanism, when it has an effect, most often implements “garbage collection.” At the same time it will also ensure that failed registrations are (re-)established.

When MRP Participants are connected by a shared or virtual shared medium, protocol performance is improved and the effect on other protocols using the LAN minimized if the Participants transmit at different times, avoiding a multicast storm and allowing some to optimize their transmissions based on messages received. A request for a transmit opportunity starts a randomized join timer, with a maximum value chosen to ensure successful reregistration(s) within a Leave time period (10.7.4), and the transmit opportunity is offered when the timer expires. If more messages are to be sent than can fit in a single PDU, a further transmit opportunity is requested.

When two MRP Participants are connected by a point-to-point medium or service instance delaying MRPDU transmission provides no benefit. In bridged networks it is desirable to transmit without delay, minimizing the denial of service that might occur while registration changes propagate after reconfiguration, and maximizing the benefit from using protocols such as RSTP and MSTP. When `operPointToPointMAC` (6.4.3) is TRUE, transmit opportunities are scheduled immediately on request, subject to rate limiting (10.7.4).

Use of point-to-point service connecting at most two Participants allows further protocol optimization, e.g., receipt of an In message does acknowledge registration. However not all LAN media support reliable determination of point-to-point status, particularly if nonstandard bridge-like devices are present. When operating in point-to-point mode, MRP avoids behavior that could cause failure, potentially continuous rapid exchanges of messages, or flapping registrations if there are more than two Participants. This standard permits simple point-to-point subset implementations of MRP, but these will successfully operate on shared media—albeit at reduced efficiency.

NOTE 5—IEEE Std 802.1AE [B7] (MAC Security) can ensure that there are at most two communicating Participants.

It is also possible to simplify an MRP Participant that only wishes to make declarations; for example, for an end station that uses MMRP (10.9) to declare a need to receive group addressed frames, but is not a source of such frames, and therefore does not need to support source pruning by registering declarations from other Participants. Such an Applicant-Only Participant does not implement the Registrar or LeaveAll state machines, never sends LeaveAll, Empty, or JoinEmpty messages (which would elicit unnecessary Joins from its peer Participants), and does not implement the administrative controls defined in 10.7.2 and 10.7.3. The following four types of MRP implementation conform to this standard:

- Full Participant
- Full Participant, point-to-point subset
- Applicant-Only Participant
- Applicant-Only point-to-point subset, also referred to as the Simple-Applicant Participant

While Simple-Applicant and Full Participant point-to-point subset implementations can operate on shared media, their initial Join and Leave messages are not suppressed. Significant additional, and unnecessary, traffic can result from attaching several such implementations to the same shared medium. Devices that do not perform registration should use an Applicant-Only Participant rather than a Simple-Applicant Participant.

NOTE 6—at the time of the development of this standard the LAN MACs most commonly used were point-to-point but the use of virtual shared media was increasing.

10.7 Protocol specification

The operation of MRP as executed by the MRP Attribute Declaration (MAD, 10.2) component of an MRP Participant is represented by the following state machines:

- a) A per-Attribute Applicant state machine (10.7.7)
- b) A per-Attribute Registrar state machine (10.7.8)
- c) A LeaveAll state machine for the Participant as a whole (10.7.9)
- d) A PeriodicTransmission state machine for the Participant as a whole (10.7.10)

These state machines are specified as compact state tables, and make use of the following:

- e) Notational conventions and abbreviations for protocol events, actions, and timer operations (10.7.1)
- f) Registrar Administrative Controls (10.7.2)
- g) Applicant Administrative Controls (10.7.3)
- h) Protocol timers (10.7.4)
- i) Protocol event definitions (10.7.5)
- j) Protocol action definitions (10.7.6)

A Full Participant implements the complete Applicant state machine (Table 10-3) and the Registrar state machine (Table 10-4) for each Attribute declared, registered, or tracked, together with a single instance of the LeaveAll state machine (Table 10-5) and the PeriodicTransmission state machine (Table 10-6).

The point-to-point subset of the Full Participant implements the same state machines, but omits certain Applicant state machine states and actions as specified by Table 10-3.

An Applicant-Only Participant implements the Applicant state machine, with the omission of certain states and actions as specified by Table 10-3, for each Attribute declared, registered, or tracked, together with a single instance of the PeriodicTransmission state machine (Table 10-6).

The point-to-point subset of the Applicant-Only Participant (the Simple-Applicant Participant) implements the same state machines, but omits certain Applicant state machine states and actions as specified by Table 10-3.

NOTE—Conceptually, per-Attribute state is maintained for all possible values of all Attribute types that are defined for a given application; however, in real implementations of MRP, it is likely that the range of possible Attribute values in some applications will preclude this, and the implementation will limit the state to those Attribute values in which the Participant has an immediate interest, either as a Member or as a likely future Member.

Timer values, their relationships, and default values are described in 10.7.11 and Table 10-7, protocol management statistics accumulated by each MAD component in 10.7.12, and interoperability considerations for potentially misordering networks in 10.7.13.

The encoding of MRP messages in MRPUUs is specified in 10.8, which also specifies the parsing and checks applied to received PDUs.

10.7.1 Notational conventions and abbreviations

The following conventions are used in the abbreviations used in this subclause:

rXXX	receive PDU XXX
sXXX	send PDU XXX
txXXX	transmit opportunity

XXX!	state machine event
!XXX	“Not XXX”; i.e., logical NOT applied to the condition XXX

The following abbreviations are used in the state machine descriptions. For their meaning, see 10.7.5 and 10.7.6.

Protocol events:

Begin!	Initialize state machine (10.7.5.1)
New!	A new declaration (10.7.5.4)
Join!	Declaration without signaling new registration (10.7.5.5)
Lv!	Withdraw a declaration (10.7.5.6)
tx!	Transmission opportunity without a LeaveAll (10.7.5.7)
txLA!	Transmission opportunity with a LeaveAll (10.7.5.8)
txLAF!	Transmission opportunity with a LeaveAll, and with no room (Full) (10.7.5.9)
rNew!	receive New message (10.7.5.14)
rJoinIn!	receive JoinIn message (10.7.5.15)
rIn!	receive In message (10.7.5.18)
rJoinMt!	receive JoinEmpty message (10.7.5.16)
rMt!	receive Empty message (10.7.5.19)
rLv!	receive Leave message (10.7.5.17)
rLA!	receive a LeaveAll message (10.7.5.20)
Flush!	Port role changes from Root Port or Alternate Port to Designated Port (10.7.5.2)
Re-Declare!	Port role changes from Designated to Root Port or Alternate Port (10.7.5.3)
periodic!	A periodic transmission event occurs (10.7.5.10)
leavetimer!	leavetimer has expired (10.7.5.21)
leavealltimer!	leavealltimer has expired (10.7.5.22)
periodictimer!	periodictimer has expired (10.7.5.23)

Protocol actions:

New	send a New indication to MAP and the MRP application (10.7.6.12)
Join	send a Join indication to MAP and the MRP application (10.7.6.13)
Lv	send a Lv indication to MAP and the MRP application (10.7.6.14)
sN	send a New message (10.7.6.2)
sJ	send a JoinIn or JoinMT message (10.7.6.3)
sL	send a Lv message (10.7.6.4)
s	send an In or an Empty message (10.7.6.5)
[s]	send an In or an Empty message, if required for optimization of the encoding (10.7.6.5)
[sL]	send a Lv message, if required for optimization of the encoding (10.7.6.4)
[sJ]	send a Join message, if required for optimization of the encoding (10.7.6.3)
sLA	send a Leave All message (10.7.6.6)
periodic	Periodic transmission event (10.7.6.7).
leavetimer	Leave period timer (10.7.4.2)
leavealltimer	Leave All period timer (10.7.4.3)
periodictimer	Periodic Transmission timer (10.7.4.4)
-x-	Inapplicable event/state combination. No action or state transition occurs in this case.

Timers are used in the state machine descriptions in order to cause actions to be taken after defined time periods have elapsed. The following terminology is used in the state machine descriptions to define timer states and the actions that can be performed upon them:

- a) A timer is said to be *running* if the most recent action to be performed upon it was a *start*.

- b) A running timer is said to have *expired* when the time period associated with the timer has elapsed since the most recent start action took place.
- c) A timer is said to be *stopped* if it has expired or if the most recent action to be performed upon it was a *stop* action.
- d) A *start* action sets a stopped timer to the running state, and associates a time period with the timer. This time period supersedes any periods that might have been associated with the timer by previous start events.
- e) A *stop* action sets a timer to the stopped state.

The following abbreviations are used for the state names in the state tables and state diagrams:

Registrar states (see 10.6):

IN	In
LV	Leaving
MT	Empty

Applicant and Simple-Applicant states (see 10.6):

VO	Very anxious Observer
VP	Very anxious Passive
VN	Very anxious New
AN	Anxious New
AA	Anxious Active
QA	Quiet Active
LA	Leaving Active
AO	Anxious Observer
QO	Quiet Observer
AP	Anxious Passive
QP	Quiet Passive
LO	Leaving Observer

10.7.2 Registrar Administrative Controls

Associated with each instance of the Registrar state machines are *Registrar Administrative Control* parameters. These parameters allow administrative control to be exercised over the registration state of each Attribute value, and hence, via the propagation mechanism provided by MAP, allow control to be exercised over the propagation of declarations.

- a) *Normal Registration*. The Registrar responds to incoming MRP messages as specified by Table 10-4.
- b) *Registration Fixed*. The Registrar ignores all MRP messages, and remains IN (registered).
- c) *Registration Forbidden*. The Registrar ignores all MRP messages, and remains MT (unregistered).

The default value of this parameter is *Normal Registration*.

If the value of this parameter is *Registration Fixed* or *Registration Forbidden*, In and JoinIn messages are sent rather than Empty or JoinEmpty messages.

NOTE—The Registrar Administrative Controls are realized by means of the contents of the Port Map parameters of static entries in the Filtering Database for all MRP applications. In the case of MMRP, the static entries concerned are Static Filtering Entries (8.8.1); in the case of MVRP, the static entries concerned are Static VLAN Registration Entries (8.8.2). The contents of the Port Map parameters in static entries can be modified by means of the management operations defined in Clause 12. In the absence of such control information for a given attribute, the default value “Normal Registration” is assumed.

10.7.3 Applicant Administrative Controls

An overall control parameter for each Applicant state machine, the *Applicant Administrative Control*, determines whether or not the Applicant state machine participates in MRP exchanges.

- a) *Normal Participant*. The state machine participates normally in MRP exchanges.
- b) *Non-Participant*. The state machine does not send any MRP messages.

The default value of this parameter is Normal Participant.

NOTE 1—The Applicant Administrative Control parameters can be modified for any MRP application by means of the management operations defined in Clause 12. In the absence of such information for a given attribute, the default value “Normal Participant” is assumed.

NOTE 2—The Applicant Administrative Control parameters can be set per attribute type (see 12.9.2).

10.7.4 Protocol timers

10.7.4.1 jointimer

The Join Period Timer, jointimer, controls the interval between transmit opportunities that are applied to the Applicant state machine. An instance of this timer is required on a per-Port, per-MRP Participant basis. The value of JoinTime used to initialize this timer is determined in accordance with 10.7.11.

10.7.4.2 leavetimer

The Leave Period Timer, leavetimer, controls the period of time that the Registrar state machine will wait in the LV state before transitioning to the MT state. An instance of the timer is required for each state machine that is in the LV state. The Leave Period Timer is set to the value LeaveTime when it is started; LeaveTime is defined in Table 10-7.

10.7.4.3 leavealltimer

The Leave All Period Timer, leavealltimer, controls the frequency with which the LeaveAll state machine generates LeaveAll PDUs. The timer is required on a per-Port, per-MRP Participant basis. The Leave All Period Timer is set to a random value, T, in the range $\text{LeaveAllTime} < T < 1.5 \times \text{LeaveAllTime}$ when it is started. LeaveAllTime is defined in Table 10-7.

10.7.4.4 periodictimer

The Periodic Transmission timer, periodictimer, controls the frequency with which the PeriodicTransmission state machine generates periodic! events. The timer is required on a per-Port basis. The Periodic Transmission timer is set to one second when it is started.

10.7.5 Protocol event definitions

Unless stated otherwise in these event definitions, MRPDU reception in a Bridge can occur through all Ports of a Bridge, and events generated as a result of such reception affect only those state machines that are associated with the Port through which the PDU was received.

10.7.5.1 Begin!

The state machine is initialized or reinitialized.

10.7.5.2 Flush!

A Flush! event signals to the Registrar state machine that there is a need to rapidly deregister information on the Port associated with the state machine as a result of a topology change that has occurred in the network topology that supports the propagation of MRP information. If the network topology is maintained by means of the Spanning Tree state machines, then, for the set of Registrar state machines associated with a given Port and Spanning Tree instance, this event is generated when the Port Role changes from either Root Port or Alternate Port to Designated Port.

When a Flush! event occurs for a given Port and Spanning Tree instance, a leavealltimer! event (10.7.5.22) is also signaled to the LeaveAll state machine for that Port and Spanning Tree instance.

10.7.5.3 Re-declare!

A Re-declare! event signals to the Applicant and Registrar state machines that there is a need to rapidly redeclare registered information on the Port associated with the state machines as a result of a topology change that has occurred in the network topology that supports the propagation of MRP information. If the network topology is maintained by means of the Spanning Tree state machines, then, for the set of Applicant and Registrar state machines associated with a given Port and Spanning Tree instance, this event is generated when the Port Role changes from Designated Port to either Root Port or Alternate Port.

10.7.5.4 New!

A new declaration is made. The event is deemed to have occurred if the MAD Service User issues a MAD_Join.request service primitive for the Attribute instance associated with that state machine, indicating a new declaration.

10.7.5.5 Join!

A declaration is made. The event is deemed to have occurred if the MAD Service User issues a MAD_Join.request service primitive for the Attribute instance associated with that state machine.

10.7.5.6 Lv!

A declaration is withdrawn. The event is deemed to have occurred if the MAD Service User issues a MAD_Leave.request service primitive for the Attribute instance associated with that state machine.

10.7.5.7 tx!

A transmission opportunity occurs, without a LeaveAll being signaled by the LeaveAll state machine.

NOTE—The tx! event is modified by the behavior of the LeaveAll state machine. If the LeaveAll state machine has signaled LeaveAll, then tx! is modified to txLA! (see 10.7.5.8).

10.7.5.8 txLA!

A transmission opportunity occurs, with a LeaveAll being signaled by the LeaveAll state machine.

10.7.5.9 txLAF!

A transmission opportunity occurs, with a LeaveAll being signaled by the LeaveAll state machine, and with no room available in the PDU.

10.7.5.10 periodic!

This event indicates to the Applicant state machine that the timer used to stimulate periodic transmission has expired.

10.7.5.11 periodicEnabled!

This event indicates to the Periodic Transmission state machine that it has been enabled by management action (12.9.3).

10.7.5.12 periodicDisabled!

This event indicates to the Periodic Transmission state machine that it has been disabled by management action (12.9.3).

10.7.5.13 Message reception events

For an instance of the Applicant state machine or the Registrar state machine, a message reception event is deemed to have occurred if an MRPDU (10.8) is received, and the following conditions are true:

- a) The PDU was addressed to the MRP application address (Table 10-1) and had an EtherType (Table 10-2) in accordance with that of the MRP application associated with the state machine.
- b) The PDU contains a Message (10.8.1) in which the Attribute Type is the type associated with the state machine.
- c) The Message contains a VectorAttribute (10.8.1.2) where the range defined by the FirstValue and NumberOfValues includes the attribute value associated with the state machine.

The specific type of message reception event is determined by the event value (10.8.2.5) associated with the state machine. The possible message reception event types are specified in 10.7.5.14 through 10.7.5.20.

10.7.5.14 rNew!

For an instance of the Applicant or Registrar state machine, a message reception event (10.7.5.13) occurs, and the event value (10.8.2.5) associated with the state machine specifies the New message.

10.7.5.15 rJoinIn!

For an instance of the Applicant or Registrar state machine, a message reception event (10.7.5.13) occurs, and the event value (10.8.2.5) associated with the state machine specifies the JoinIn message.

10.7.5.16 rJoinMt!

For an instance of the Applicant or Registrar state machine, a message reception event (10.7.5.13) occurs, and the event value (10.8.2.5) associated with the state machine specifies the JoinMt message.

10.7.5.17 rLv!

For an instance of the Applicant or Registrar state machine, a message reception event (10.7.5.13) occurs, and the event value (10.8.2.5) associated with the state machine specifies the Lv message.

10.7.5.18 rIn!

For an instance of the Applicant or Registrar state machine, a message reception event (10.7.5.13) occurs, and the event value (10.8.2.5) associated with the state machine specifies the In message.

10.7.5.19 rMt!

For an instance of the Applicant or Registrar state machine, a message reception event (10.7.5.13) occurs, and the event value (10.8.2.5) associated with the state machine specifies the Mt message.

10.7.5.20 rLA!

For an instance of the Applicant state machine, the Registrar state machine, or the LeaveAll state machine, the rLA! event is deemed to have occurred if either:

- a) The LeaveAll state machine associated with that instance of the Applicant or Registrar state machine performs the sLA action (10.7.6.6); or
- b) An MRPDU (10.8) is received, and the following conditions are true:
 - 1) The PDU's destination MAC address and EtherType are in accordance with the definition of the MRP application associated with the state machine.
 - 2) The PDU contains a Message (10.8.1) in which the Attribute Type is the type associated with the state machine.
 - 3) The Message contains a LeaveAllEvent (10.8.2.6) that carries the value LeaveAll.

NOTE—The LeaveAll state machine operates on a per-application (not per-Attribute Type) basis, but the LeaveAll message operates on a per-Attribute Type basis. Hence, when the LeaveAll state machine issues a LeaveAll, it must generate a LeaveAll Attribute for each Attribute Type supported by the application concerned.

10.7.5.21 leavetimer!

For an instance of the Registrar state machine, the leavetimer! event is deemed to have occurred when the leavetimer associated with that state machine expires.

10.7.5.22 leavealltimer!

For an instance of the LeaveAll state machine, the leavealltimer! event is deemed to have occurred either when the leavealltimer associated with that state machine expires or when a Flush! event (10.7.5.2) occurs for the Port and Spanning Tree instance associated with that state machine.

10.7.5.23 periodictimer!

For an instance of the PeriodicTransmission state machine, the periodictimer! event is deemed to have occurred when the periodictimer associated with that state machine expires.

10.7.6 Protocol Action definitions**10.7.6.1 MRPDU transmission actions**

Unless stated otherwise in these action definitions, MRPDU transmission as a result of the operation of a state machine in a Bridge occurs only through the Port associated with that state machine, and only if that Port is in the Forwarding state. MRPDUs shall be transmitted using the destination MAC Address and EtherType value defined by the MRP application associated with the state machine.

When the action specifies the transmission of attribute state information, an MRPDU, formatted as defined in 10.8.1, is transmitted, such that:

- a) The PDU contains a Message (10.8.1.2) that carries an Attribute Type (10.8.2.2) that corresponds to the type of attribute associated with the state machine concerned.

- b) The Message contains a VectorAttribute (10.8.2.10) in which the FirstValue (10.8.2.7) and NumberOfValues (10.8.2.8) defines a range of attribute values that includes the attribute value associated with the state machine.
- c) The Vector (10.8.2.10) encodes an AttributeEvent value in the vector position corresponding to the attribute value for this state machine. The choice of AttributeEvent value is determined by the specific transmission action, as defined below.

In the case of LeaveAll transmission actions, the LeaveAll event value specified by the transmission action is encoded in the NumberOfValues field of the VectorAttribute.

10.7.6.2 sN

The AttributeEvent value New is encoded in the Vector as specified in 10.7.6.1.

10.7.6.3 sJ, [sJ]

If the Registrar state is IN, then the AttributeEvent value JoinIn is encoded in the Vector as specified in 10.7.6.1.

If the Registrar state is MT or LV, then the AttributeEvent value JoinMt is encoded in the Vector as specified in 10.7.6.1.

NOTE—The [sJ] variant indicates that the action is only necessary in cases where transmitting the value, rather than terminating a vector and starting a new one, makes for more optimal encoding; i.e., transmitting the value is not necessary for correct protocol operation.

10.7.6.4 sL

The AttributeEvent value Lv is encoded in the Vector as specified in 10.7.6.1.

10.7.6.5 s, [s]

If the Registrar state is IN, then the AttributeEvent value In is encoded in the Vector as specified in 10.7.6.1.

If the Registrar state is MT or LV, then the AttributeEvent value Mt is encoded in the Vector as specified in 10.7.6.1.

NOTE—In the [s] variant, this value is transmitted only if its inclusion makes for more optimal encoding; i.e., transmitting the value is not necessary for correct protocol operation.

10.7.6.6 sLA

The LeaveAll event value specified by the transmission action is encoded in the NumberOfValues field of the VectorAttribute as specified in 10.7.6.1.

The sLA action also gives rise to a rLA! event (10.7.5.20) against all instances of the Applicant state machine, or the Registrar state machine, associated with the MRP participant.

10.7.6.7 periodic

Causes a periodic! event (10.7.5.10) against all Applicant state machines associated with the participant.

10.7.6.8 Start leavetimer

Causes leavetimer to be started, in accordance with the definition of the timer in 10.7.4.2.

10.7.6.9 Stop leavetimer

Causes leavetimer to be stopped.

10.7.6.10 Start leavealltimer

Causes leavealltimer to be started, in accordance with the definition of the timer in 10.7.4.3.

10.7.6.11 Start periodictimer

Causes periodictimer to be started, in accordance with the definition of the timer in 10.7.4.4.

10.7.6.12 New

This action causes a MAD_Join.indication primitive to be issued to the MAD Service User, indicating the Attribute instance corresponding to the state machine concerned.

10.7.6.13 Join

This action causes a MAD_Join.indication primitive to be issued to the MAD Service User, indicating the Attribute instance corresponding to the state machine concerned.

10.7.6.14 Lv

This action causes a MAD_Leave.indication primitive to be issued to the MAD Service User, indicating the Attribute instance corresponding to the state machine concerned.

10.7.7 Applicant state machine

A full MRP Participant maintains a single instance of the Applicant state machine (Table 10-3) for each Attribute value for which the Participant needs to maintain state information.

Table 10-3—Applicant state table

		STATE											
		VO ¹¹	VP ⁶	VN ⁶	AN ⁶	AA ⁶	QA	LA ⁶	AO ^{3,11}	QO ^{3,11}	AP ^{3,6}	QP ³	LO ⁶
EVENT	Begin!	—	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO
	New!	VN	VN	—	—	VN	VN	VN	VN	VN	VN	VN	VN
	Join!	VP	—	—	—	—	—	AA	AP	QP	—	—	VP
	Lv!	—	VO	LA	LA	LA	LA	—	—	—	AO	QO	—
	rNew!	—	—	—	—	—	—	—	—	—	—	—	—
	rJoinIn!	AO ⁴	AP ⁴	—	—	QA	—	—	QO	—	QP	—	—
	rIn!	—	—	—	—	QA ⁵	—	—	—	—	—	—	—
	rJoinMt! rMt!	—	—	—	—	—	AA	—	—	AO	—	AP	VO
	rLv! rLA! Re-declare!	LO ¹	—	—	VN	VP ⁹	VP ⁹	— ¹⁰	LO ¹	LO ¹	VP	VP	—
	periodic!	—	—	—	—	—	AA	—	—	—	—	AP	—
	tx! ⁷	[s] —	sJ AA	sN AN	sN QA ⁸	sJ QA	[sJ] —	sL VO	[s] —	[s] —	sJ QA	[s] —	s VO
	txLA! ²	[s] LO	s AA	sN AN	sN QA	sJ QA	sJ —	[s] LO	[s] LO	[s] LO	sJ QA	sJ QA	[s] —
	txLAF! ²	LO	VP	VN	VN	VP	VP	LO	LO	LO	VP	VP	—

Notes to the table:

¹Applicant-Only participants exclude the LO state, and transition to VO.

²These events do not occur for Applicant-Only participants.

³Point-to-point subset participants exclude the AO, QO, AP, and QP states.

⁴Ignored (no transition) if point-to-point subset or if operPointToPointMAC is TRUE.

⁵Ignored (no transition) if operPointToPointMAC is FALSE. See MRP Design Notes below.

⁶Request opportunity to transmit on entry to VN, AN, AA, LA, VP, AP, and LO states.

⁷If the MRPDU is full and cannot convey a required message there is no change of state and an additional transmit opportunity is requested if that has not been done already.

⁸QA if the Registrar is IN, and AA otherwise. See MRP Design Notes below.

MRP design notes:

⁵On shared media the receipt of In does not confirm registration by all Participants, and the In could have been sent by an Applicant-Only participant.

⁸Since New messages do not convey registrar state, a Leave could have been received without an Empty or JoinEmpty prompt being sent, the transition to AA guards against loss of that Leave by another Applicant.

⁹The design accepts a small possibility of a continued registration (after rLv! if a Lv! occurs before a further Join is sent) in return for not accumulating many Active participants when Join!s and Lv!s are frequent. rLv! processing is deliberately not optimized for point-to-point.

¹⁰If a Leave has been received, the Registrar for the transmitting participant is very probably IN, as this Applicant has not yet sent a Leave, so the pending Leave is required. The small savings from avoiding transmission of Leaves pending on receipt of LeaveAlls does not merit distinguishing the rLv! and rLA! cases.

¹¹The VO, AO, and QO states represent states where the attribute is neither being declared by the Participant nor being registered by any other station on the LAN. In implementations where dynamic creation and discarding of state machines is desirable, the state machine can be discarded when in any of these states, pending a future requirement to declare or register that attribute value.

10.7.8 Registrar state machine

A full MRP Participant maintains a single instance of this state machine for each Attribute value that is currently registered, or that the Registrar state machine is in the process of deregistering.

NOTE—As with the Applicant, state information is conceptually maintained for all possible values of all Attribute types that are defined for a given application; however, in real implementations of MRP, it is likely that the range of possible Attribute values in some applications will preclude this, and the implementation will limit the state to those Attribute values in which the Participant has an immediate interest. In the case of simple devices that have no interest in what other Participants have registered, it may be appropriate for that device to ignore Registrar operation altogether.

The detailed operation of this state machine is described in Table 10-4.

Table 10-4—Registrar state table

		STATE		
		IN	LV	MT
EVENT	Begin!	MT	MT	MT
	rNew!	New IN	New Stop leavetimer IN	New IN
	rJoinIn! rJoinMt!	IN	Stop leavetimer IN	Join IN
	rLv! rLA! txLA! Re-declare!	Start leavetimer LV	-x-	-x-
	Flush!	MT	Lv MT	MT
	leavetimer!	-x-	Lv MT	MT

10.7.9 LeaveAll state machine

A single LeaveAll state machine exists for each full MRP Participant. Leave All messages generated by this state machine also generate LeaveAll events against all the Applicant and Registrar state machines associated with that Participant and Port; hence, LeaveAll generation is treated by those state machines in the same way as reception of a LeaveAll message from an external source.

The detailed operation of this state machine is described in Table 10-5.

10.7.10 PeriodicTransmission state machine

A single PeriodicTransmission state machine exists for each Port. Periodic Transmission events are generated on a regular basis, against all Applicant state machines that are associated with that Port.

The detailed operation of this state machine is described in Table 10-6.

Table 10-5—LeaveAll state table

		STATE	
		Active ^a	Passive
EVENT	Begin!	Start leavealltimer Passive	Start leavealltimer Passive
	tx!	sLA Passive	-x-
	rLA!	Start leavealltimer Passive	Start leavealltimer Passive
	leavealltimer!	Start leavealltimer Active	Start leavealltimer Active

^aRequest opportunity to transmit on entry to the Active state

Table 10-6—PeriodicTransmission state table

		STATE	
		Active	Passive
EVENT	Begin!	Start periodictimer Active	Start periodictimer Active
	periodicEnabled!	-x-	Start periodictimer Active
	periodicDisabled!	Passive	-x-
	periodictimer!	Start periodictimer periodic Active	-x-

10.7.11 Timer values

MRP *correctness* is not critically dependent on the values of protocol timers. However the protocol operates more efficiently, and with less likelihood of unwanted de-registrations, if the following relationships are maintained between timer values used by peer Participants:

- a) LeaveTime should be at least twice the maximum JoinTime, plus six times the timer resolution, to allow reregistration after a Leave or LeaveAll even if a message is lost.
- b) To minimize the volume of rejoining traffic generated following a LeaveAll, the value chosen for LeaveAllTime should be large relative to LeaveTime.

The default timer values (Table 10-7) maintain these relationships. They may be modified on a per-Port basis by means of the management functionality defined in Clause 12.

NOTE—The default values for the MRP timers are independent of media access method or data rate. This is a deliberate choice, made in the interests of maximizing the “plug and play” characteristics of the protocol.

Implementation of the timers for MRP shall be based on a timer resolution of 1 centisecond or less.

Table 10-7—MRP timer parameter values

Parameter	Value (centiseconds)
JoinTime	20
LeaveTime	60–100
LeaveAllTime	1000

If operPointToPointMAC (6.6.3) is TRUE, a request for a transmit opportunity should result in such an opportunity as soon as is practicable, given other system constraints, and shall occur within the value specified for JoinTime subject to not more than three such transmission opportunities occurring in any period of $1.5 \times \text{JoinTime}$.

If operPointToPointMAC is FALSE, and there is no pending request, a transmit opportunity shall occur at a time value randomized between 0 and JoinTime seconds. These provisions shall apply even if a point-to-point subset Applicant has been implemented.

10.7.12 Operational reporting and statistics

10.7.12.1 Failure to register

Each MRP Participant maintains a count of the number of times that it has received a registration request, but has failed to register the attribute concerned due to lack of space in the Filtering Database to record the registration. The value of this count may be examined by management (see 12.9.2.1.3).

NOTE—Further action to be taken on such events is a matter for implementation choice.

10.7.12.2 Peer tracking

An implementation may support the ability to record against each Registrar state machine the MAC Address of the originator of the MRPDU that caused the most recent state change for that state machine (see 12.9.3.1.3).

10.7.13 Interoperability considerations

Correct operation of MRP for a given MRP application requires that protocol exchanges among a given set of communicating MRP Participants maintain sequentiality; i.e., that Participant A cannot receive MRPDU B (generated as a consequence of Participant B receiving MRPDU A) before Participant A has received MRPDU A. In circumstances where the Participants concerned are all attached to the same LAN, such sequentiality is ensured. However, if a set of MRP Participants communicates via an intervening Bridge that does not implement that MRP application (or does not implement MRP at all), the sequentiality constraints expressed in 8.6.6, 8.6.7, and 8.6.8 are insufficient to guarantee the correct operation of MRP. In order for the correct sequencing of PDUs to be maintained through such a Bridge, the following constraint must be met:

If MRPDU A is received on Port X, and is due to be forwarded on Ports Y and Z, and subsequent to being forwarded on Y, MRPDU B is received on Port Y for forwarding on Port Z, then forwarding of B cannot precede A on Port Z.

NOTE—This expresses a stronger sequencing constraint for multicast frames than is stated in 8.6, but a weaker constraint than was required for conformance to IEEE Std 802.1D, 1993 Edition [B9].

The consequence of failure to meet this constraint is that the users of a given MRP application may experience an increased incidence of loss of registration. Therefore, it is inadvisable to construct LAN configurations involving forwarding of MRPDUs through intervening Bridges if those Bridges do not meet the constraint expressed previously.

10.8 Structure and encoding of MRP Protocol Data Units

This subclause describes the generic structure and encoding of the MRP Protocol Data Units (MRPDUs) exchanged between all MRP Participants. The structure and encoding of elements that are specific to the operation of the MRP applications are defined by the applications themselves.

Each MRPDU identifies the MRP application by which it was generated, and to which it is being transmitted. Bridges that receive MRPDUs identified as belonging to an MRP application that they do not support shall forward such PDUs on all other Ports that are in a Forwarding state.

NOTE 1—If MRP is used to support an application that can operate in any MAP Context other than 0 (the Base Spanning Tree), the application specification describes how that context is identified in protocol exchanges.

Each MRPDU carries one or more MRP messages, each of which identify one or more MRP events (e.g., Join, Leave, LeaveAll) and the attribute class(es) and value(s) to which each event applies. A given MRP Participant shall process MRPDUs in the order in which they are received, and shall process the MRP Messages in a PDU in the order in which they were put into the Data Link Service Data Unit (DLSDU).

NOTE 2—Any messages generated as a consequence of state machine responses to an sLA action and its associated LeaveAll events will be put into the DLSDU after the LeaveAll message(s), or into a later DLSDU.

10.8.1 Structure

10.8.1.1 Transmission and representation of octets

All MRPDUs consist of an integral number of octets, numbered starting from 1 and increasing in the order that they are put into a Data Link Service Data Unit (DLSDU). The bits in each octet are numbered from 1 to 8, where 1 is the low-order bit.

When consecutive octets are used to represent a binary number, the lower octet number has the most significant value.

When the encoding of (an element of) an MRPDU is represented using a diagram in this clause, the following representations are used:

- a) Octet 1 is shown toward the top of the page, higher numbered octets being toward the bottom.
- b) Where more than one octet appears on a given line, octets are shown with the lowest numbered octet to the left, higher numbered octets being to the right.
- c) Within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

10.8.1.2 Structure definition

MRP makes use of an EtherType value as the means of identifying the MRP application that has transmitted, and that will receive, a given MRPDU. Table 10-2 lists the set of MRP applications that are defined, and the EtherType values that correspond to them.

A protocol version field is included in the structure of the MRPDU, in order to provide the ability to identify future enhancements to MRP applications.

MRPDUs exchanged according to the protocol specified in this clause shall have the following structure:

- a) The first octet contains the *ProtocolVersion*.
- b) Following the Protocol Version are one or more *Messages*. The last element in the PDU is an *EndMark*.
- c) Each Message consists of an *AttributeType*, an *AttributeLength*, and an *AttributeList*, in that order.
- d) An Attribute List consists of one or more *VectorAttributes*. The last element in the AttributeList is an *EndMark*.
- e) A VectorAttribute consists of a *VectorHeader*, a *FirstValue*, and a *Vector*, in that order. The VectorHeader is able to encode both a *LeaveAllEvent* and the number of attribute events encoded in the vector.
- f) If the end of an MRPDU is encountered before an EndMark is reached, then processing of the PDU is terminated as if an EndMark had been reached.

The following Backus-Naur Form (BNF) productions give the formal description of the MRPDU structure:

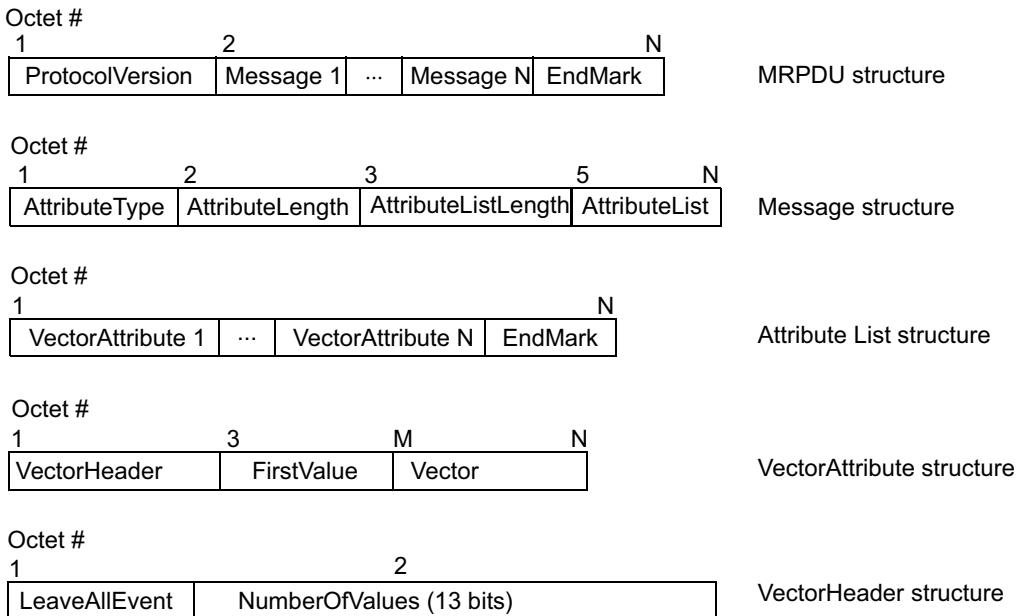
```

MRPDU ::= ProtocolVersion, Message {, Message}, EndMark
ProtocolVersion BYTE ::= Defined by the specific MRP application
Message ::= AttributeType, AttributeLength[, AttributeListLength], AttributeList
AttributeType BYTE ::= Nonzero integer defined by the specific MRP application
AttributeLength BYTE ::= Nonzero integer defined by the specific MRP application
AttributeListLength SHORT ::= Nonzero integer defined by the specific MRP application
AttributeList ::= VectorAttribute {, VectorAttribute}, EndMark
VectorAttribute ::= VectorHeader, FirstValue, Vector
VectorHeader SHORT ::= (LeaveAllEvent * 8192) + NumberOfValues
FirstValue ::= Defined by the specific MRP application
Vector ::= ThreePackedEvents {, ThreePackedEvents}
      [, FourPackedEvents {, FourPackedEvents}]
ThreePackedEvents BYTE ::= (((((AttributeEvent *6) + AttributeEvent) *6) + AttributeEvent)
AttributeEvent BYTE ::= New | JoinIn | In | JoinMt | Mt | Lv
FourPackedEvents BYTE ::= ((FourPackedType *64) + (FourPackedType *16)
      + (FourPackedType *4) + FourPackedType)
FourPackedType BYTE ::= Integer defined by the specific MRP application
LeaveAllEvent BYTE ::= NullLeaveAllEvent | LeaveAll
NumberOfValues SHORT ::= Number of events encoded in the vector
EndMark SHORT ::= 0x0000 | End of PDU
NullLeaveAllEvent ::= 0
LeaveAll ::= 1
New ::= 0
JoinIn ::= 1
In ::= 2
JoinMt ::= 3
Mt ::= 4
Lv ::= 5

```

The parameters carried in MRPDUs, as identified in this structure definition, shall be encoded as specified in 10.8.2.

Figure 10-5 illustrates the structure of the MRPDU and its components.

**Figure 10-5—Format of the major components of an MRPDU**

10.8.2 Encoding of MRPDU parameters

10.8.2.1 Encoding of ProtocolVersion

A ProtocolVersion shall be encoded in a single octet, taken to represent an unsigned binary number. It takes a value that is determined by the application concerned.

NOTE—The handling of protocol version information is defined in 10.8.3.5.

10.8.2.2 Encoding of AttributeType

An AttributeType shall be encoded as a single octet, taken to represent an unsigned binary number. The AttributeType identifies the type of Attribute to which the message applies. The range of values that can be taken by the AttributeType, and the meanings of those values, are defined by the application concerned. The value 0 is reserved, and shall not be used as an AttributeType by any MRP application. MRP applications may otherwise allocate meanings to any set of values of AttributeType in the range 1 through 255.

The definition of AttributeType values in the application associates the following with each AttributeType value defined:

- a) The size, in octets, of attribute values, and how they are encoded.
- b) Any restrictions that are placed upon the range of attribute values that the vector is permitted to represent.

10.8.2.3 Encoding of AttributeLength

An AttributeLength shall be encoded as a single octet, taken to represent an unsigned binary number. The AttributeLength indicates the length, in octets, of the FirstValue field (10.8.2.7) for the Attribute to which the message applies.

10.8.2.4 Encoding of AttributeListLength

An AttributeListLength shall be encoded as two octets, taken to represent an unsigned binary number. The AttributeListLength indicates the length, in octets, of the AttributeList field for the Attribute to which the message applies. This field is not present in all MRPDUs. Specifically, MMRPDUs and MVRPDUs do not use this field, whereas MSRPDUs do use this field.

10.8.2.5 Encoding of AttributeEvent

An AttributeEvent shall be encoded as an unsigned decimal number in the range 0 through 5. The permitted values and meanings of the AttributeEvent are as follows:

- 0: New operator
- 1: JoinIn operator
- 2: In operator
- 3: JoinMt operator
- 4: Mt operator
- 5: Lv operator

Further values of AttributeEvent are reserved.

The AttributeEvent is interpreted on receipt as a MAD event to be applied to the state machine for the Attribute defined by the AttributeType and AttributeValue to which the AttributeEvent relates.

10.8.2.6 Encoding of LeaveAllEvent

A LeaveAllEvent shall be encoded as an unsigned binary number. The permitted values and meanings of LeaveAllEvent are as follows:

- 0: NullLeaveAllEvent operator
- 1: LeaveAll operator

Further values of LeaveAllEvent are reserved.

The LeaveAllEvent is interpreted on receipt as a MAD Leave All event to be applied to the state machines for all Attributes of the type defined by the AttributeType field.

The value NullLeaveAllEvent signifies that there is no Leave All event to process, and is included purely for encoding efficiency in the vector attribute structures. Receipt of this value does not cause any event to be applied to any state machine.

10.8.2.7 Encoding of FirstValue

A FirstValue is encoded in N octets, taken to be an unsigned binary number, in accordance with the specification for the AttributeType defined by the MRP application concerned. The length, in octets, of the FirstValue field is specified by the value contained in the AttributeLength field (10.8.2.3). When NumberOfValues is greater than one (1) FirstValue is incremented in a way that is defined by each MRP application. For example MMRP simply increments FirstValue by adding the number one (1) to it, whereas MSRP adds one (1) to multiple fields within the FirstValue for each increment. Throughout this specification FirstValue incremented by one is denoted as FirstValue + 1, incrementing by two is denoted as FirstValue + 2, etc.

10.8.2.8 Encoding of VectorHeader

The VectorHeader is used to encode both the value of the LeaveAllEvent (10.8.2.6) and the NumberOfValues, the number of AttributeEvent values encoded in a Vector (10.8.2.10). The VectorHeader is taken to be an unsigned binary number, encoded in two octets, as follows:

- a) The value of the LeaveAllEvent is multiplied by 8192.
- b) The resulting number is added to NumberOfValues.

The range of values that NumberOfValues can take is restricted, such that the following are true:

- c) The size of the Vector that is defined by this number will fit in the available space in the PDU.
- d) Incrementing FirstValue the number of times specified in NumberOfValues does not exceed the permitted numeric range of FirstValue as defined for the application concerned.
- e) The number of AttributeEvent values is nonzero, and does not exceed 8191.

10.8.2.9 Encoding of EndMark

An EndMark shall be encoded as two octets, taken to represent an unsigned binary number. It takes the numeric value 0.

Further values of EndMark are reserved and shall not be used.

NOTE—As defined by the MRPDU structure definition in 10.8.1, if the end of the MRPDU is encountered, this is taken to be an End Mark from the point of view of processing the PDU contents.

10.8.2.10 Encoding of Vector

10.8.2.10.1 Encoding of Vector ThreePackedEvents

The Vector is encoded as zero or more 8-bit values, each containing a numeric value, ThreePackedEvents, derived from three packed numeric values, each of which represents an AttributeEvent, in the range 0 through 5.

As can be seen from the BNF definition of ThreePackedEvents, each 8-bit value is derived by successively adding an event value and multiplying the result by 6. In order to facilitate the subsequent description, the event values are numbered from *first* to *third*, as follows:

```
ThreePackedEvents BYTE ::= (((((firstAttributeEvent) *6) + secondAttributeEvent)  
*6) + thirdAttributeEvent)
```

The NumberOfValues field in the VectorHeader of the VectorAttribute determines the number of 8-bit ThreePackedEvents values, *E*, that will be present in the vector; hence *E* is determined by dividing NumberOfValues by 3 and rounding any noninteger answer up to the nearest larger integer.

The FirstValue field of the VectorAttribute determines which of the originator's state machines the *first* AttributeEvent value in the first ThreePackedEvents value relates to. The *second* AttributeEvent value in the first ThreePackedEvents value corresponds to the state machine identified by (FirstValue + 1), and the *third* AttributeEvent value in the first ThreePackedEvents value corresponds to the state machine identified by (FirstValue + 2), and so on, through subsequent packed values.

Where the NumberOfValues field carries a value that is not a multiple of 3, there will be either one or two AttributeEvent values packed in the final ThreePackedEvents that are ignored; these values are encoded as the numeric value 0 on transmission and are ignored on receipt.

NOTE—If NumberOfValues is zero, there will be no ThreePackedEvents encoded in the vector.

10.8.2.10.2 Encoding of Vector FourPackedEvents

The Vector is encoded as zero or more 8-bit values, each containing a numeric value, FourPackedEvents, derived from four packed numeric values, each of which represent a FourPackedType, in the range 0 through 3. The FourPackedTypes are defined by each MRP application that uses FourPackedEvents. Note that not all MRP applications use FourPackedEvents.

As can be seen from the BNF definition of FourPackedEvents, each 8-bit value is derived by successively adding a FourPackedType value and multiplying the result by 4. In order to facilitate the subsequent description, the event values are numbered from first to fourth, as follows:

```
FourPackedEvents BYTE ::= ((firstFourPackedType * 64) + (secondFourPackedType * 16) +
                           (thirdFourPackedType * 4) + (fourthFourPackedType))
```

The NumberOfValues field in the VectorHeader of the VectorAttribute determines the number of 8-bit FourPackedEvents values, E, that will be present in the vector; hence E is determined by dividing NumberOfValues by 4 and rounding any noninteger answer up to the nearest larger integer.

The FirstValue field of the VectorAttribute determines which of the originator's state machines the first FourPackedType value in the first FourPackedEvents value relates to. The second FourPackedType value in the first FourPackedEvents value corresponds to the state machine identified by (FirstValue + 1). The third FourPackedType value in the first FourPackedEvents value corresponds to the state machine identified by (FirstValue + 2), and the fourth FourPackedType value in the first FourPackedEvents value corresponds to the state machine identified by (FirstValue + 3), and so on, through subsequent packed values.

Where the NumberOfValues field carries a value that is not a multiple of 4, there will be one, two or three FourPackedType values packed in the final FourPackedEvent that are ignored; these values are encoded as the numeric value 0 on transmission and are ignored on receipt.

NOTE— If NumberOfValues is zero, there will be no FourPackedEvents encoded in the vector.

10.8.3 Packing and parsing MRPDUs

The use of the End Mark (10.8.2.9) to signal the end of an AttributeList and the end of an MRPDU, and the fact that the (physical) end of the PDU is interpreted as an End Mark, simplifies the requirements both for packing information into MRPDUs and for correctly interpreting that information on receipt.

10.8.3.1 Packing

Successive Messages are packed into the MRPDU, and within each Message, successive VectorAttributes are packed into each Message, until the end of the PDU is encountered or there are no more VectorAttributes to pack at that time. The following cases can occur:

- a) The PDU has sufficient room for all the VectorAttributes that require to be transmitted at that time to be packed. In this case, the PDU is transmitted, and subsequent PDUs are transmitted when there are further VectorAttributes to transmit.
- b) The PDU has enough room for the first N VectorAttributes that require to be transmitted at that time to be packed. In this case, the PDU is transmitted, and the next N VectorAttributes are encoded in a subsequent PDU.

10.8.3.2 Handling of received MRPDUs

Received MRPDUs, i.e., PDUs that are constructed in accordance with the MRPDU format defined in 10.8, that carry a destination MAC address and MRP EtherType value as specified for use by a defined MRP application, are processed as follows:

- a) In Bridges and end stations that support the operation of the MRP application concerned, all such PDUs shall be submitted to the MRP Participant associated with the reception Port for further processing.
- b) In Bridges that do not support the operation of the MRP application concerned, all such PDUs shall be submitted to the Forwarding Process.

10.8.3.3 Discarding badly formed MRPDUs

An MRP Participant that receives an MRPDU shall discard that PDU if the PDU is not formatted according to the MRPDU format defined in 10.8.

10.8.3.4 Parsing

Successive Messages, and within each Message, successive VectorAttributes, are unpacked from the PDU. If this process terminates because the end of the PDU is reached, then the end of the PDU is taken to signal termination both of the current AttributeList and the overall PDU. The following two cases can occur:

- a) The last VectorAttribute to be unpacked was complete. In this case, the VectorAttribute is processed normally, and processing of the PDU terminates.
- b) The last VectorAttribute to be unpacked was incomplete. In this case, the entire PDU is discarded, and processing of the PDU terminates.

10.8.3.5 Handling of protocol versions

In order to ensure compatibility with previous protocol versions, while allowing the protocol to be extended in the future, the following requirements shall be met by the implementation:

- a) MRPDUs that carry a protocol version lower than the protocol version implemented shall be interpreted according to the definition corresponding to the protocol version carried in the PDU.
- b) MRPDUs that carry a protocol version equal to or higher than the protocol version implemented shall be interpreted according to the definition corresponding to the protocol version implemented.
- c) Where the MRPDU carries a higher protocol version than the version implemented, then:
 - 1) If a Message is encountered in which the AttributeType is not recognized, then that Message is discarded. This is achieved by discarding the successive VectorAttributes in the AttributeList until either an EndMark or the end of the PDU is reached. If an EndMark is reached, processing continues with the next Message.
 - 2) If a VectorAttribute is encountered in which the AttributeEvent is not recognized for the AttributeType concerned, then the VectorAttribute is discarded and processing continues with the next VectorAttribute, if the end of the PDU has not been reached.

10.9 Multiple MAC Registration Protocol (MMRP)—Purpose

MMRP provides a mechanism that allows end stations and MAC Bridges to dynamically register (and subsequently, deregister) Group membership and individual MAC address information with the Bridges attached to the same LAN, and disseminates that information across all the Bridges that support Extended Filtering Services in the Bridged Local Area Network. The operation of MMRP relies upon the services provided by MRP.

The information registered, deregistered, and disseminated via MMRP is in the following forms:

- a) *Group membership information.* This indicates the presence of MMRP participants that are members of a particular Group (or Groups), and carries the group MAC address(es) associated with the Group(s). The exchange of specific Group membership information can result in the creation or updating of MAC Address Registration Entries in the Filtering Database to indicate the Port(s) and VID(s) of the VLAN(s) on which members of the Group(s) have been registered. The structure of these entries is described in 8.8.4.
- b) *Group service requirement information.* This indicates that one or more MMRP participants require Forward All Groups or Forward Unregistered Groups to be the default Group filtering behavior (see 6.16.7 and 8.8.6).
- c) *Individual MAC address information.*

Registration of Group membership information makes Bridges aware that frames destined for the group MAC address concerned should only be forwarded in the direction of the registered members of the Group. Therefore, forwarding of frames destined for the address associated with that Group occurs only on Ports on which such membership registration has been received.

Registration of Group service requirement information makes the Bridges aware that Ports that can forward frames in the direction from which the information has been received should modify their default Group forwarding behavior in accordance with the service requirement expressed.

NOTE—Modification of default Group forwarding behavior allows Bridge Ports to accommodate MMRP-unaware devices in the Bridged Local Area Network by forwarding frames destined for unregistered group MAC addresses.

The operation of MMRP can result in

- d) The propagation of Group membership information and Group service requirement information, and consequent creation, updating, or deletion of MAC Address Registration Entries in the Filtering Databases of all Bridges in the network that support Extended Filtering Services.
- e) Consequent changes to the Group filtering behavior of such Bridges.

In VLAN Bridges, MMRP operates only when the Bridge Filtering Mode is set to Extended Filtering Mode. Bridges that are unable to operate in Extended Filtering Mode, or have been set to operate in Basic Filtering Mode, are transparent with respect to MMRP exchanges, and forward any MRPDUs destined for the MMRP application through all Ports that are in Forwarding.

NOTE—An MMRP user is not prevented from registering an individual MAC address associated with a location different from its actual location. Such registration could result in denial of service to the station associated with that MAC Address and, possibly, the interception of associated traffic by an unintended recipient. Such consequences of the misuse of individual MAC address registration can be prevented by using MMRP individual MAC address registration only in secure network environments (i.e., only in networks providing perimeter security).

10.10 Model of operation

MMRP defines an *MRP application* that provides the extended filtering services defined in 6.16.5 and 6.16.7. To this end, MMRP makes use of

- a) The declaration and propagation services offered by *MRP Attribute Distribution* (MAD; 10.2 and 10.3,) and *MRP Attribute Propagation* (MAP; 10.2 and 10.3) to declare and propagate Group membership, Group service requirement, and individual MAC address information within the Bridged Local Area Network.

- b) The registration services offered by MAD (10.2 and 10.3) to allow Group membership, Group service requirement, and individual MAC address information to control the frame filtering behavior of participating devices.

Figure 10-6 illustrates the architecture of MMRP in the case of a two-Port Bridge and an end station, for a given VLAN Context. Where MMRP is used in multiple VLAN Contexts, an instance of the MMRP Participant exists for each VLAN context.

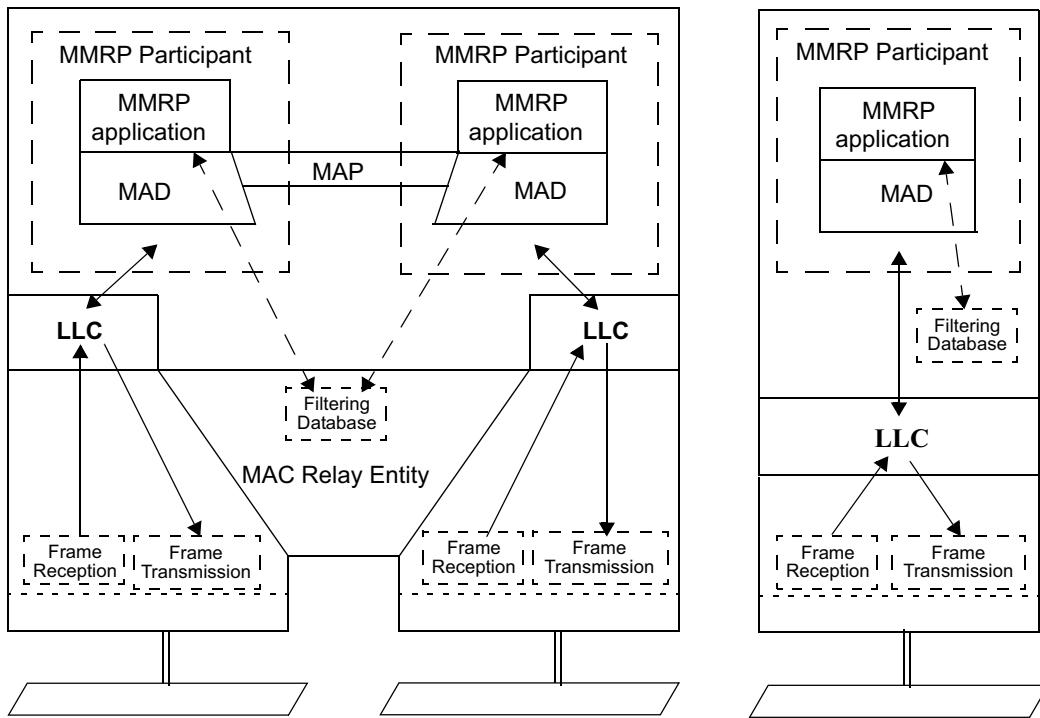


Figure 10-6—Operation of MMRP for a single VLAN Context

As shown in the diagram, the MMRP Participant consists of the following components:

- c) The MMRP application, described in 10.12
- d) MRP Attribute Propagation (MAP), described in 10.3
- e) MRP Attribute Declaration, described in 10.2

10.10.1 Propagation of Group Membership information

The Forwarding Process uses the MAC Address Registration Entries in the Filtering Databases to ensure that frames are transmitted only through those Bridge Ports necessary to reach LANs to which Group members are attached. Figure 10-7 illustrates the MAC Address Registration Entries created by MMRP for a single Group.

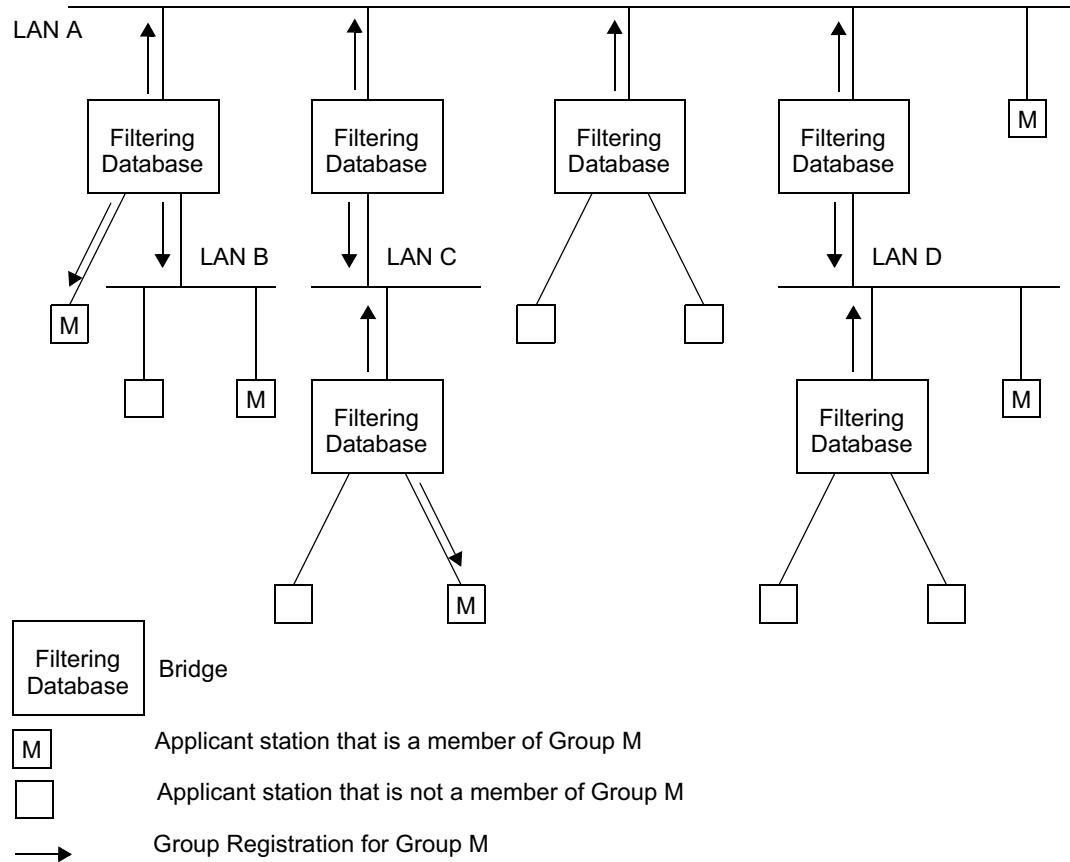


Figure 10-7—Example Directed Graph

By receiving frames from all Ports and forwarding only through Ports for which MMRP has created MAC Address Registration Entries, Bridges facilitate Group distribution mechanisms based on the concept of an Open Host Group. Any MMRP Participants (10.2) that wish to receive frames transmitted to a particular Group or Groups request membership of the Group(s) concerned. Any MAC Service user that wishes to send frames to a particular Group can do so from any point of attachment to the Bridged Local Area Network. These frames can be received on all LANs to which registered MMRP Participants are attached, but the filtering applied by Bridges ensures that frames are not transmitted on LANs that are not part of the active topology between the sources of the frames and the registered Group members. MMRP and the MAC Address Registration Entries thus restrict the frames to pruned subsets of the overall loop-free active topology.

NOTE—The term “Open Host Group” comes from the terminology introduced in the definition of the Internet Group Membership Protocol (IGMP) defined by the IETF.

MAC Service users that are sources of MAC frames destined for the Group do not have to register as members of the Group themselves unless they also wish to receive frames transmitted to the Group address by other sources.

10.10.2 Propagation of Group service requirement information

MMRP propagates Group service requirement information in the same manner as for Group registration information. If any Port in a given Bridge has a registered Group service requirement of All Groups or All Unregistered Groups (expressed in terms of the control information in Static Filtering Entries and/or MAC

Address Registration Entries with a MAC Address specification of All Groups or All Unregistered Groups), this fact is propagated on all other Ports of the Bridge, resulting in the registration of that information on Ports of adjacent Bridges. As a consequence of that registration, the default Group filtering behavior of those Ports can change in order to maintain compatibility with the service requirements expressed by the registered information, as defined in 8.8.6. This ensures that connectivity can be maintained in LANs where the service requirements of different regions of the Bridged Local Area Network differ.

NOTE—In a Bridged Local Area Network where the default Group filtering behavior is not the same for all “edge” Ports, service requirement propagation will tend to result in all “backbone” Ports switching to the highest precedence Group filtering behavior in use in the network. The precedence rules are defined in 8.8.6.

10.10.3 Source pruning

As described in 10.10.1, the operation of MMRP defines a subtree of the Spanning Tree as a result of the creation of MAC Address Registration Entries in the Filtering Databases of the Bridges. End stations are also able to make use of the Group Membership information registered via MMRP to allow them to keep track of the set of Groups for which active members currently exist and the service requirements of upstream devices. This allows end stations that are sources of frames destined for a Group to suppress the transmission of such frames, if their registered Group membership and Group service requirement information indicates that there are no valid recipients of those frames reachable via the LANs to which they are attached.

NOTE—In effect, for the purposes of frame transmission, the end station can be viewed as if it operates as a single Port Bridge, with its own default Group filtering behavior and “Filtering Database” entries updated via MMRP that tell it whether or not multicast frames that it has generated should be forwarded onto the attached LAN. In order to achieve this, it is necessary for the end station to implement both the Registrar and the Applicant functionality of MRP, as described in 10.6, 10.7.7, and 10.7.8. The Applicant-Only and Simple-Applicant Participants described in 10.6 do not contain the Registrar functionality that would be required for source pruning.

This end system behavior is known as *source pruning*. Source pruning allows MAC Service users that are sources of MAC frames destined for a number of Groups, such as server stations or routers, to avoid unnecessary flooding of traffic on their local LANs in circumstances where there are no current Group members in the network that wish to receive such traffic.

10.10.4 Use of Group service requirement registration by end stations

The ability to propagate Group service requirement information is described in this standard primarily as a means of propagating the requirements of the Bridges themselves. However, this mechanism can also be used by end stations that have requirements involving some aspect of promiscuous reception, such as Routers or network monitors. A MMRP-aware end station wishing to receive all multicast traffic can declare membership of All Groups on the LAN to which it is attached; similarly, an end station that wishes to receive unregistered multicast traffic can do so by declaring membership of All Unregistered Groups.

10.11 Default Group filtering behavior and MMRP propagation

The propagation of MMRP registrations within a VLAN Context has implications with respect to the choice of default Group filtering behavior within a Bridged LAN. As MMRP frames are transmitted only on outbound Ports that are in the Member set (8.8.10) for the VID concerned, propagation of Group registrations by a given Bridge occurs only toward regions of the Bridged LAN where that VID has been (statically or dynamically) registered. This is illustrated in Figure 10-8; dotted lines in the diagram show those regions of the LAN where propagation of registrations for Group M for VID V does not occur. Consequently, the Filtering Databases of the lower two Bridges will not contain any Dynamic MAC Address Registration Entry for Group M for VID V.

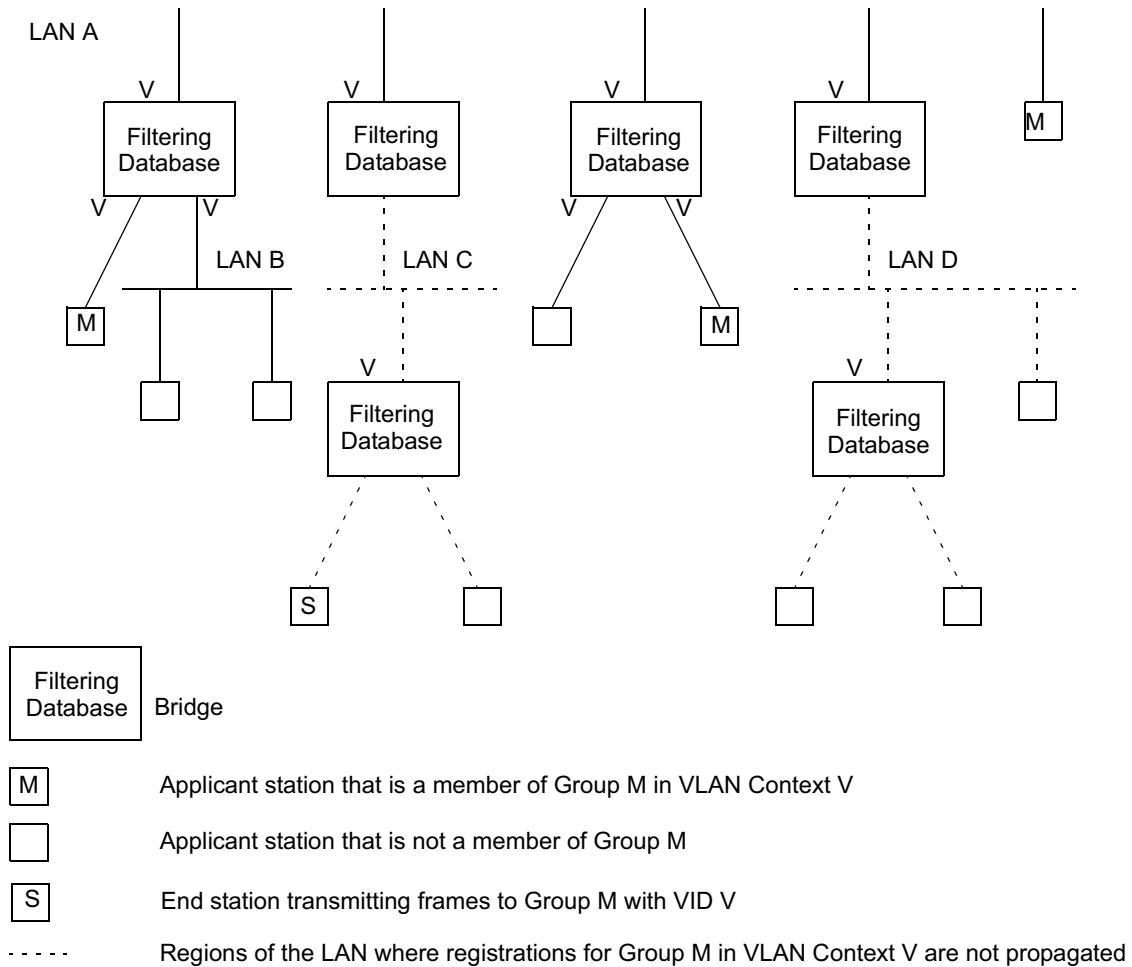


Figure 10-8—Example of MMRP propagation in a VLAN Context

The action of these two Bridges on receipt of frames, on either of their lower Ports, destined for Group M and VID V, will depend upon the Default Group Filtering Behavior adopted by their upper Ports, which are the Ports that are in the Member set for VID V. If the Default Group Filtering Behavior is either Forward All Groups or Forward Unregistered Groups, then these Bridges will forward the frames. If the Default Group Filtering Behavior is Filter Unregistered Groups, then these Bridges will filter the frames. In the scenario shown, the choice of Default Group Filtering Behavior is therefore crucial with respect to whether or not end station S, or any other station that is “outside” the VLAN, is able to send frames to members of the Group. The choice between Filter Unregistered Groups and the other default behaviors therefore has the effect of defining VLANs that are closed to external unregistered traffic (Filter Unregistered Groups) or open to external unregistered traffic (either of the other default behaviors).

10.12 Definition of the MMRP application

10.12.1 Definition of MRP elements

10.12.1.1 Use of MAP Contexts by MMRP

MMRP, as defined in this standard, operates within the set of VLAN Contexts that correspond to the VIDs that are supported by the VLAN Bridged Local Area Network. The Base Spanning Tree Context is not used for the propagation of MMRP information. The use of MMRP in any other MAP Context is outside the scope of this standard.

NOTE 1—In IEEE Std 802.1D, propagation of GMRP information takes place only in the Base Spanning Tree Context.

The MAP Context Identifier used to identify a VLAN Context shall be equal to the VID used to identify the corresponding VLAN.

The set of Ports of a Bridge defined to be part of the active topology for a given VLAN Context shall be equal to the set of Ports of a Bridge for which the following are true:

- a) The Port is a member of the *Member set* (8.8.10) for that VID; and
- b) The Port is one of the Ports of the Bridge that are part of the active topology for the spanning tree that supports that VID.

NOTE 2—The above definition applies equally to SST and MST environments. It ensures that MMRP operates in either environment, without the MMRP implementations needing to be aware of whether the VLAN Contexts that apply are all supported by the same spanning tree, as in the SST environment, or are potentially distributed across two or more spanning trees, as in the MST environment.

10.12.1.2 Context identification in MMRP

Implementations of MMRP in VLAN Bridges apply the same ingress rules (8.6.2) to received MMRPDUs that are defined for the reception Port. Therefore

- a) MMRP frames with no VLAN classification (i.e., untagged or priority-tagged MMRPDUs) are discarded if the Acceptable Frame Types parameter (6.9) for the Port is set to *Admit Only VLAN-tagged frames*. Otherwise, they are classified either according to the PVID for the Port, or as determined by protocol-based VLAN classification (6.12) if that capability is implemented.
- b) VLAN-tagged MMRP frames are classified according to the VID carried in the tag header, optionally translated using the VID translation table (6.9).
- c) If Ingress Filtering (8.6.2) is enabled, and if the Port is not in the Member set (8.8.10) for the MMRP frame's VLAN classification, then the frame is discarded.

The VLAN classification thus associated with a received MMRP frame establishes the VLAN Context for the received PDU, and identifies the MRP Participant instance to which the PDU is directed.

MMRPDUs transmitted by MMRP Participants are VLAN classified according to the VLAN Context associated with that Participant. MMRP Participants in VLAN Bridges apply the same egress rules that are defined for the transmission Port (8.6.4). Therefore

- d) MMRPDUs are transmitted through a given Port only if the Member Set for the VID concerned indicates that the VID is registered on that Port.
- e) MMRPDUs are transmitted through a given Port as VLAN-tagged frames or as untagged frames in accordance with the state of the Untagged Set (8.8.10) for that Port and for the VID concerned.

Where VLAN-tagged frames are transmitted, the VID field of the tag header carries the VLAN Context Identifier value, optionally translated using the VID translation table (6.9).

10.12.1.3 MMRP application address

The group MAC address used as the destination address for MRPDUs destined for MMRP Participants shall be the group MAC address identified in Table 10-1 as “Customer and Provider Bridge MMRP address.”

10.12.1.4 MMRP application EtherType

The EtherType used for MRPDUs destined for MMRP Participants shall be the MMRP EtherType identified in Table 10-2.

10.12.1.5 MMRP ProtocolVersion

The ProtocolVersion for the version of MMRP defined in this standard takes the hexadecimal value 0x00.

10.12.1.6 MMRP AttributeType definitions

MMRP defines two AttributeTypes (10.8.2.2) that are carried in MRP exchanges, as follows:

- a) The Service Requirement Vector Attribute Type.
- b) The MAC Vector Attribute Type.

Attributes identified by the Service Requirement Vector Attribute Type are instances of VectorAttributes (10.8.1), used to identify values of Group service requirements. The value of AttributeType used to identify the Service Requirement Vector Attribute Type in MRPDUs (10.8.2.2) shall be 1.

Attributes identified by the MAC Vector Attribute Type are instances of VectorAttributes (10.8.1), used to identify a sequence of values of MAC Addresses. The value of AttributeType used to identify the MAC Vector Attribute Type in MRPDUs (10.8.2.2) shall be 2.

10.12.1.7 MMRP FirstValue definitions

The FirstValue field (10.8.2.7) in instances of the MAC Vector Attribute Type shall be encoded in MRPDUs as six octets, each taken to represent an unsigned binary number. The octets are derived from the Hexadecimal Representation of an EUI-48 MAC Address (defined in IEEE Std 802) as follows:

- a) Each two-digit hexadecimal numeral in the Hexadecimal Representation is taken to represent an unsigned hexadecimal value, in the normal way, i.e., the rightmost digit of each numeral represents the least significant digit of the value, the leftmost digit is the most significant.
- b) The first octet of the attribute value encoding is derived from the left-most hexadecimal value in the Hexadecimal Representation of the MAC Address. The LSB of the octet (bit 1) is assigned the LSB of the hexadecimal value, the second significant bit is assigned the value of the second significant bit of the hexadecimal value, and so on.
- c) The second through sixth octets of the encoding are similarly assigned the value of the second through sixth hexadecimal values in the Hexadecimal Representation of the MAC Address.

FirstValue+1 is defined as adding 1 to FirstValue. There are no restrictions on the range of values that can be represented in this data type.

The FirstValue field in instances of the Service Requirement Attribute Type shall be encoded in MRPDUs (10.8.2.7) as a single octet, taken to represent an unsigned binary number. Only two values of this type are defined:

- d) All Groups shall be encoded as the value 0.
- e) All Unregistered Groups shall be encoded as the value 1.

The remaining possible values (2 through 255) are reserved.

10.12.1.8 MMRP AttributeLength definitions

The AttributeLength field (10.8.2.3) in instances of the MAC Vector Attribute Type shall be encoded in MRPDUs (10.8) as an unsigned binary number, equal to the value 0x06.

The AttributeLength field (10.8.2.3) in instances of the Service Requirement Vector Attribute Type shall be encoded in MRPDUs (10.8) as an unsigned binary number, equal to the value 0x01.

10.12.1.9 MMRP AttributeListLength definitions

The AttributeListLength field (10.8.2.4) is not present in the MMRPDUs.

10.12.1.10 MMRP Vector definitions

The ThreePackedEvent vectors are encoded as defined in 10.8.2.10.1.

The FourPackedEvent vectors (10.8.2.10.2) are not present in the MMRPDUs.

10.12.2 Provision and support of Extended Filtering Services

10.12.2.1 Initiating MMRP registration and de-registration

The MMRP application element of an MRP Participant provides the dynamic registration and de-registration services defined in 6.16.7.1, as follows:

On receipt of a REGISTER_MAC_ADDRESS service primitive, the MMRP Participant issues a MAD_Join.request. The attribute_type parameter of the request carries the value of the MAC Vector Attribute Type (10.12.1.6) and the attribute_value parameter carries the value of the MAC_ADDRESS parameter of the service primitive.

On receipt of a Deregister_MAC_ADDRESS service primitive, the MMRP Participant issues a MAD_Leave.request. The attribute_type parameter of the request carries the value of the MAC Vector Attribute Type (10.12.1.6) and the attribute_value parameter carries the value of the MAC_ADDRESS parameter of the service primitive.

On receipt of a REGISTER_SERVICE_REQUIREMENT service primitive, the MMRP Participant issues a MAD_Join.request. The attribute_type parameter of the request carries the value of the Service Requirement Vector Attribute Type (10.12.1.6) and the attribute_value parameter carries the value of the REQUIREMENT_SPECIFICATION parameter of the service primitive.

On receipt of a Deregister_SERVICE_REQUIREMENT service primitive, the MMRP Participant issues a MAD_Leave.request. The attribute_type parameter of the request carries the value of the Service Requirement Vector Attribute Type (10.12.1.6) and the attribute_value parameter carries the value of the REQUIREMENT_SPECIFICATION parameter of the service primitive.

NOTE—If the EISS is supported, the Extended Filtering service primitives issued by the MAC Service user should include a VID parameter in addition to the MAC_ADDRESS or REQUIREMENT_SPECIFICATION parameters, in order to identify the MMRP Participant associated with the primitive.

10.12.2.2 Registration and de-registration events

The MMRP application element of an MRP Participant responds to registration and de-registration events signaled by MAD as follows:

On receipt of a MAD_Join.indication, the MMRP application element specifies the Port associated with the MMRP Participant as Forwarding in the Port Map field of the MAC Address Registration Entry (8.8.4) for the MAC Address specification carried in the attribute_value parameter and the VID associated with the MAP Context. If such a MAC Address Registration Entry does not exist in the Filtering Database, a new MAC Address Registration Entry is created.

Creation of new MAC Address Registration Entries may be restricted by the Restricted_MAC_Address_Registration control (10.12.2.3). If this control is TRUE, creation of a new dynamic entry is permitted only if there is a Static Filtering Entry for the MAC Address with a Registrar Administrative Control value of Normal Registration.

NOTE—Group Membership and Group service requirement information can be recorded in the Filtering Database by means of MAC address Registration Entries (see 8.8.4). In the case of Group Membership Information, the MAC Address specification in the MAC address Registration Entry is a Group MAC Address. In the case of Group service requirement information, the MAC Address specification is either “All Group Addresses” or “All Unregistered Group Addresses.”

On receipt of a MAD_Leave.indication, the MMRP application element specifies the Port associated with the MMRP Participant as Filtering in the Port Map field of the MAC Address Registration Entry (8.8.4) for the MAC Address specification carried in the attribute_value parameter and the VID associated with the MAP Context. If such a Filtering Database entry does not exist in the Filtering Database, then the indication is ignored. If setting that Port to Filtering results in there being no Ports in the Port Map specified as Forwarding (i.e., all MMRP members are deregistered), then that MAC Address Registration Entry is removed from the Filtering Database.

10.12.2.3 Administrative controls

The provision of static control over the registration state of the state machines associated with the MMRP application is achieved by means of the Registrar Administrative Control parameters associated with the operation of MRP (10.7.2). These parameters are represented in the Filtering Database by the information held in Static Filtering Entries (8.8.1). If no Static Filtering Entry exists for a given MAC Address specification, the value of the Registrar Administrative Control parameter for the corresponding attribute value is Normal Registration for all Ports of the Bridge.

The initial state of the Permanent Database (i.e., the state of the Permanent Database in a Bridge that has not been otherwise configured by management action) includes a Static Filtering Entry with a MAC Address specification of All Groups, in which the Port Map indicates Registration Fixed. This Static Filtering Entry will have the effect of determining the default Group filtering behavior of all Ports of the Bridge to be Forward All Groups. This Permanent Database entry may be deleted or updated by management action.

NOTE—This specification of an initial Static Filtering Entry means that operation using Forward Unregistered Groups or Filter Unregistered Groups requires a conscious action on the part of the network manager or administrator.

Where management capability is implemented, the Registrar Administrative Control parameters can be applied and modified by means of the management functionality defined in 12.7.

The provision of static control over the ability of Applicant state machines to participate in protocol exchanges is achieved by means of the Applicant Administrative Control parameters associated with the operation of MRP (10.7.3). Where management capability is implemented, the Applicant Administrative Control parameters can be applied and modified by means of the management functionality defined in Clause 12.

Further administrative control over dynamic MAC Address registration may be achieved, if supported, by means of a per-Port Restricted_MAC_Address_Registration control parameter. If the value of this control is TRUE for a given Port, the creation or modification of Dynamic MAC Address Registration Entries as a result of MMRP exchanges on that Port shall be restricted only to those MAC addresses for which Static Filtering Entries exist in which the Registrar Administrative Control value is Normal Registration. If the value of the Restricted_MAC_Address_Registration control is FALSE, dynamic MAC Address registration is not so restricted. Where management capability is implemented, the value of the Restricted_MAC_Address_Registration control can be manipulated by means of the management functionality defined in Clause 12. If management of this parameter is not supported, the value of this parameter shall be FALSE for all Ports.

10.12.3 Use of “new” declaration capability

MMRP does not make use of the “new” declaration capability.

10.12.4 Attribute value support requirements

Implementations of MMRP shall maintain state information for all attribute values that support the Group service requirement registration (10.12.1.7).

Implementations of MMRP shall be capable of supporting any attribute value in the range of possible values that can be registered using Group membership and individual MAC address registration (10.12.1.7); however, the maximum number of attribute values for which the implementation is able to maintain current state information is an implementation decision. The number of values that the implementation can support shall be stated in the PICS.

11. VLAN topology management

The egress rules (8.6.4) defined for the Forwarding Process in VLAN Bridges rely on the existence of configuration information for each VID that defines the set of Ports of the Bridge through which one or more members are reachable. This set of Ports is known as the Member Set (8.8.9), and its membership is determined by the presence or absence of configuration information in the Filtering Database, in the form of Static and Dynamic VLAN Registration Entries (8.8.2, 8.8.5).

Reliable operation of the VLAN infrastructure requires VLAN membership information held in the Filtering Database to be maintained in a consistent manner across all VLAN-aware Bridges in the Bridged Local Area Network, in order to ensure that frames destined for end station(s) on a given VLAN can be correctly delivered, regardless of where in the Bridged Local Area Network the frame is generated. Maintenance of this information by end stations that are sources of VLAN-tagged frames can allow such stations to suppress transmission of such frames if no members exist for the VLAN concerned.

This standard defines the following mechanisms that allow VLAN membership information to be configured:

- a) Dynamic configuration and distribution of VLAN membership information by means of the Multiple VLAN Registration Protocol (MVRP), as described in 11.2.
- b) Static configuration of VLAN membership information via Management mechanisms, as described in Clause 12, which allow configuration of Static VLAN Registration Entries.

These mechanisms provide for the configuration of VLAN membership information as a result of the following:

- c) Dynamic registration actions taken by end stations or Bridges in the Bridged Local Area Network.
- d) Administrative actions.

11.1 Static and dynamic VLAN configuration

The combined functionality provided by the ability to configure Static VLAN Registration Entries in the Filtering Database, coupled with the use of the Restricted_VLAN_Registration control (11.2.3.2.3) and the ability of MVRP to dynamically create and update Dynamic VLAN Registration Entries, offers the following possibilities with respect to how VIDs are configured on a given Port:

- a) *Static configuration only.* The management facilities described in Clause 12 are used to establish precisely which VIDs have this Port in their Member set, and the MVRP management controls are used to disable the operation of MVRP on that Port. Hence, any use of MVRP by devices reachable via that Port is ignored, and the Member set for all VIDs can therefore only be determined by means of static entries in the Filtering Database.
- b) *Dynamic configuration only.* The operation of MVRP is relied upon to establish Dynamic VLAN Registration Entries that will dynamically reflect which VIDs are registered on the Port, their contents changing as the configuration of the network changes. The MVRP management controls are set to enable the operation of MVRP on that Port.
- c) *Combined static and dynamic configuration.* The static configuration mechanisms are used in order to configure some VLAN membership information; for other VIDs, MVRP is relied upon to establish the configuration. The MVRP management controls are set to enable the operation of MVRP on that Port.

All of the previous approaches are supported by the mechanisms defined in this standard, and each approach is applicable in different circumstances. For example:

- d) Use of static configuration may be appropriate on Ports where the configuration of the attached devices is fixed, or where the network administrator wishes to establish an administrative boundary outside of which any MVRP registration information is to be ignored. For example, it might be desirable for all Ports serving end user devices to be statically configured in order to ensure that particular end users have access only to VLANs identified by particular VIDs.
- e) Use of dynamic configuration may be appropriate on Ports where the VLAN configuration is inherently dynamic; where users of particular VIDs can connect to the network via different Ports on an ad hoc basis, or where it is desirable to allow dynamic reconfiguration in the face of Spanning Tree topology changes. In particular, if the “core” of the Virtual Bridged Local Area Network contains redundant paths that are pruned by the operation of Spanning Tree, then it is desirable for Bridge Ports that form the core network to be dynamically configured.
- f) Use of both static and dynamic configuration can be appropriate on Ports where it is desirable to place restrictions on the configuration of some VIDs, while maintaining the flexibility of dynamic registration for others. For example, on Ports serving mobile end user devices, this would maintain the benefits of dynamic VLAN registration from the point of view of traffic reduction, while still allowing administrative control over some VIDs via that Port.

11.2 Multiple VLAN Registration Protocol

The Multiple VLAN Registration Protocol (MVRP) defines an *MRP application* that provides the VLAN registration service defined in 11.2.2. MVRP makes use of MRP Attribute Declaration (MAD) and MRP Attribute Propagation (MAP), which provide the common state machine descriptions and the common attribute propagation mechanisms defined for use in MRP-based applications. The MRP architecture, MAD, and MAP are defined in Clause 10.

MVRP provides a mechanism for dynamic maintenance of the contents of Dynamic VLAN Registration Entries for each VID, and for propagating the information they contain to other Bridges. This information allows MVRP-aware devices to dynamically establish and update their knowledge of the set of VIDs associated with VLANs that currently have active members, and through which Ports those members can be reached.

11.2.1 MVRP overview

The operation of MVRP is closely similar to the operation of MMRP (10.9), which is used for registering Group membership and individual MAC address information. The primary differences are as follows:

- a) The attribute values carried by the protocol are 12-bit VID values, rather than Group service requirement information and EUI-48 MAC Addresses.
- b) The act of registering/deregistering a VID affects the contents of Dynamic Registration Entries (8.8.5), rather than the contents of MAC Address Registration Entries (8.8.4).
- c) In an SST environment, there is a single MVRP Participant per port, as opposed to one MMRP Participant per VID per port, and the MVRP Participants all operate in a single MAP Context.
- d) In an MST environment, there is again a single MVRP Participant per port, but each MVRP Participant operates in multiple MAP Contexts.

MVRP allows both end stations and Bridges in a Bridged Local Area Network to issue and revoke declarations relating to membership of VLANs. The effect of issuing such a declaration is that each MVRP Participant that receives the declaration will create or update a Dynamic VLAN Registration Entry in the Filtering Database to indicate that VID is registered on the reception Port. Subsequently, if all Participants on a LAN that had an interest in a given VID revoke their declarations, the Port attached to that LAN is set to Unregistered in the Dynamic VLAN Registration Entry for that VID by each MVRP Participant attached to that LAN.

Figure 11-1 illustrates the architecture of MVRP in the case of a two-Port Bridge and an end station.

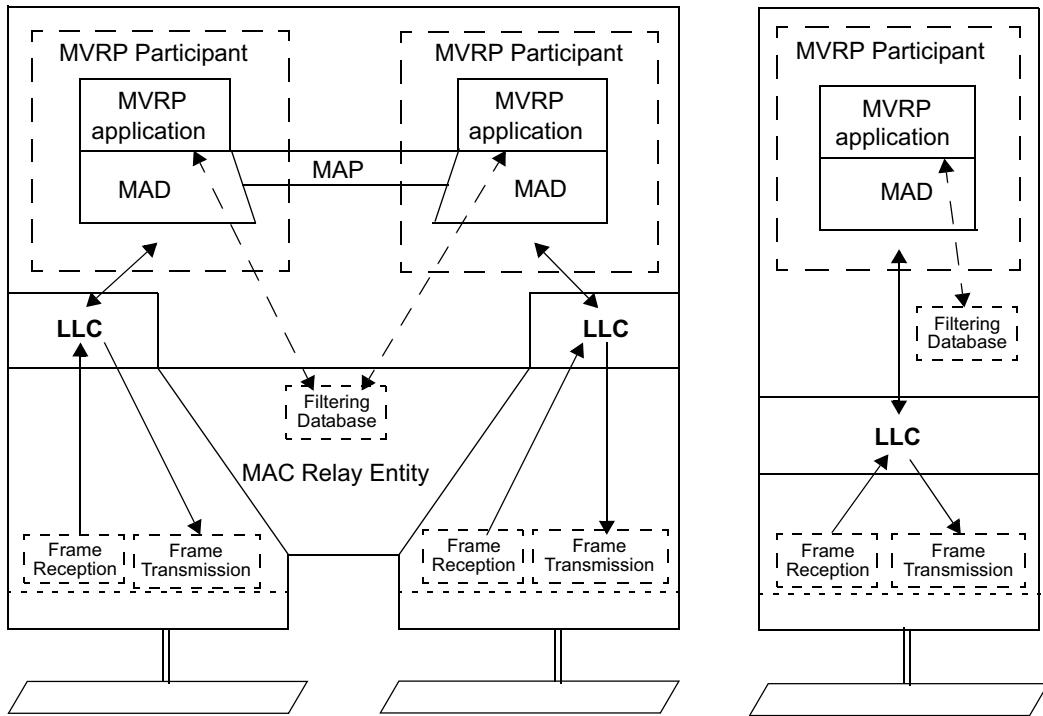


Figure 11-1—Operation of MVRP

As shown in the diagram, the MVRP Participant consists of the following components:

- e) The MVRP application, described in 11.2.3
- f) MRP Attribute Propagation (MAP), described in 10.3
- g) MRP Attribute Declaration, described in 10.2

11.2.1.1 Behavior of end stations

VLAN-aware end stations participate in MVRP activity, as appropriate for the set of VIDs of VLANs of which they are currently members. MVRP provides a way for such an end station to ensure that the VIDs of VLAN(s) of which it is a member are registered for each Port on any LAN to which the end station is attached. MVRP also provides for that VID information to be propagated across the Spanning Tree to other VLAN-aware devices, as described in 11.2.1.2.

Incoming VLAN membership information (from all other devices on the same LAN) allows such end stations to “source prune” (i.e., discard at source; see 10.10.3) any traffic destined for VLANs that currently have no other members in the Bridged Local Area Network, thus avoiding the generation of unnecessary traffic on their local LANs. This is illustrated in Figure 11-1 by a Filtering Database shown as being present in the end station.

NOTE—Non-VLAN-aware end stations have no need to register VLAN membership via MVRP; indeed, this would be impossible for them to achieve if truly VLAN-unaware, as they would have no knowledge of the set of VLANs in which they participate. Their VLAN registration requirements are taken care of by means of the configuration of PVIDs (and possibly other VLAN classification mechanisms) and the propagation of registered VIDs by the Bridges.

11.2.1.2 Behavior of Bridges

VLAN-aware Bridges register and propagate VLAN memberships on all Bridge Ports that are part of the active topology of the underlying Spanning Tree(s). Incoming VID registration and de-registration information is used to update the Dynamic VLAN Registration Entries associated with each VID. Any changes in the state of registration of a given VID on a given Port are propagated on Ports that are part of the active topology of the underlying Spanning Tree, in order to ensure that other MVRP-aware devices in the Bridged Local Area Network update their Filtering Databases appropriately. In Bridges that support multiple Spanning Tree instances, the MST Configuration Table (12.12, 13.7) is used to determine which spanning tree instance is to be used to propagate registration information for each supported VID.

The Filtering Databases in all MVRP-aware devices are thus automatically configured such that the Port Map in the Dynamic VLAN Registration Entry for a given VID indicates that a given Port is registered if one or more members of the corresponding VLAN are reachable through the Port.

NOTE—The information that determines whether frames destined for each VID are transmitted VLAN-tagged or untagged is carried in Static VLAN Registration Entries (8.8.2); if no such entry exists for a VID, then it is assumed that frames for that VID are transmitted VLAN-tagged on all Ports. Therefore, if the configuration information held in the Filtering Database for a given VID consists only of information configured by the operation of MVRP (i.e., only a Dynamic VLAN Registration Entry), then all traffic for that VID will be VLAN-tagged on transmission.

11.2.1.3 Use of the PVID and VID Set

The initial state of the Permanent Database contains a Static VLAN Registration Entry for the Default PVID, in which the Port Map indicates Registration Fixed on all Ports. This ensures that in the default state, where the value of every PVID of each Port is the Default PVID and where the VID Set of each Port is empty, membership of the Default PVID is propagated across the Bridged Local Area Network to all other MVRP-aware devices. Subsequent management action may change both the Permanent Database and the Filtering Database in order to modify or remove this initial setting, and may change the PVID and/or VID Set value(s) on any Port of the Bridge.

NOTE—In the absence of any modification of these initial settings, this ensures that connectivity is established across the Bridged Local Area Network for the VLAN corresponding to the Default PVID.

11.2.2 VLAN registration service definition

The VLAN registration service allows MAC Service users to indicate to the MAC Service provider the set of VLANs in which they wish to participate; i.e., that the MAC Service user wishes to receive traffic destined for members of that set of VLANs. The service primitives allow the service user to

- a) Register membership of a VLAN
- b) Deregister membership of a VLAN

Provision of these services is achieved by means of MVRP and its associated procedures, as described in 11.2.3.

ES_REGISTER_VLAN_MEMBER (VID)

Indicates to the MAC Service provider that the MAC Service user wishes to receive frames destined for the VLAN identified by the VID parameter.

ES_DEREGISTER_VLAN_MEMBER (VID)

Indicates to the MAC Service provider that the MAC Service user no longer wishes to receive frames destined for the VLAN identified by the VID parameter.

The use of these services can result in the propagation of VID information across the Spanning Tree, affecting the contents of Dynamic VLAN Registration Entries (8.8.5) in Bridges and end stations in the Bridged Local Area Network, and thereby affecting the frame forwarding behavior of those Bridges and end stations.

11.2.3 Definition of the MVRP application

11.2.3.1 Definition of MRP elements

11.2.3.1.1 MAP Context for MVRP in SST environments

In an SST environment, MVRP operates in the Base Spanning Tree Context (10.3.1); i.e., MVRP operates only on the CIST. Consequently, all MVRPDUs sent and received by MVRP Participants in SST bridges are transmitted as untagged frames.

11.2.3.1.2 MAP Contexts for MVRP in MST environments

In an MST environment, MVRP operates in multiple spanning-tree contexts, one for each of the spanning trees. Each spanning-tree context consists of the ports that are in the forwarding state for the corresponding spanning tree. The MVRP Participants associated with a given spanning tree operate in the MAP context identified by that spanning tree instance. MST bridges can identify the MAP Contexts using the mappings of VID values to MSTID values (see 8.9.1). All MVRPDUs sent and received by MVRP Participants in MST bridges are transmitted as untagged frames.

11.2.3.1.3 MVRP application address

The group MAC address used as the destination address for MRPDUs destined for MVRP Participants shall be either:

- a) The group MAC address identified in Table 10-1 as “Customer Bridge MVRP address” if the MVRP Participant is implemented in a C-VLAN component; or
- b) The group MAC address identified in Table 8-1 as “Provider Bridge MVRP Address” if the MVRP Participant is implemented in an S-VLAN component.

11.2.3.1.4 MVRP application EtherType

The EtherType used for MRPDUs destined for MVRP Participants shall be the MVRP EtherType identified in Table 10-2.

11.2.3.1.5 MVRP ProtocolVersion

The ProtocolVersion for the version of MVRP defined in this standard takes the hexadecimal value 0x00.

11.2.3.1.6 MVRP AttributeType definitions

MVRP defines a single Attribute Type (10.8.2.2) that is carried in MRP exchanges, as follows:

Attributes identified by the VID Vector Attribute Type are instances of VectorAttributes (10.8.1), used to identify a sequence of values of VIDs. The value of AttributeType used to identify the VID Vector Attribute Type in MRPDUs (10.8.2.2) shall be 1.

11.2.3.1.7 MVRP FirstValue definitions

The FirstValue field in instances of the VID Vector Attribute Type shall be encoded in MRPDUs (10.8) as two octets, taken to represent an unsigned binary number, and equal to the value of the VID that is to be encoded.

The range of permitted VID values that can be encoded in the FirstValue fields in MVRP is restricted to the range 1 through 4094.

11.2.3.1.8 MVRP AttributeLength definitions

The AttributeLength field (10.12.1.8) in instances of the VID Vector Attribute Type shall be encoded in MRPDUs (10.8) as an unsigned binary number, equal to the value 0x02.

11.2.3.1.9 MVRP AttributeListLength definitions

The AttributeListLength field (10.8.2.4) is not present in the MVRPDUs.

11.2.3.1.10 MVRP Vector definitions

The ThreePackedEvent vectors are encoded as defined in 10.8.2.10.1.

The FourPackedEvent vectors (10.8.2.10.2) are not present in the MVRPDUs.

11.2.3.2 Provision and support of the VLAN registration service

11.2.3.2.1 Initiating VLAN membership declaration

The MVRP application element of a MVRP Participant provides the dynamic registration and de-registration services defined in 11.2.2, as follows:

On receipt of an ES_REGISTER_VLAN_MEMBER service primitive, the MVRP Participant issues a MAD_Join.request service primitive (10.2, 10.3). The attribute_type parameter of the request carries the value of the VID Vector Attribute Type (11.2.3.1.6) and the attribute_value parameter carries the value of the VID parameter carried in the ES_REGISTER_VLAN_MEMBER primitive.

On receipt of an ES_DEREGISTER_VLAN_MEMBER service primitive, the MVRP Participant issues a MAD_Leave.request service primitive (10.2, 10.3). The attribute_type parameter of the request carries the value of the VID Vector Attribute Type (11.2.3.1.6) and the attribute_value parameter carries the value of the VID parameter carried in the ES_REGISTER_VLAN_MEMBER primitive.

11.2.3.2.2 VLAN membership registration

The MVRP application element of a MVRP Participant responds to registration and de-registration events signaled by MAD as follows:

On receipt of a MAD_Join.indication whose attribute_type is equal to the value of the VID Vector Attribute Type (11.2.3.1.6), the MVRP application element indicates the reception Port as Registered in the Port Map of the Dynamic VLAN Registration Entry for the VID indicated by the attribute_value parameter. If no such entry exists, there is sufficient room in the Filtering Database, and the VID is within the range of values supported by the implementation (see 9.6), then an entry is created. If not, then the indication is not propagated and the registration fails.

The creation of new Dynamic VLAN Registration Entries can be restricted by use of the Restricted_VLAN_Registration control (11.2.3.2.3). If the value of this control is TRUE, then creation of a new dynamic entry is permitted only if there is a Static VLAN Registration Entry for the VID concerned, in which the Registrar Administrative Control value is Normal Registration.

On receipt of a MAD_Leave.indication whose attribute_type is equal to the value of the VID Vector Attribute Type (11.2.3.1.6), the MVRP application element indicates the reception Port as not Registered in the Port Map of the Dynamic VLAN Registration Entry for the VID indicated by the attribute_value parameter. If no such entry exists, the indication is ignored.

11.2.3.2.3 Administrative controls

The provision of static control over the declaration or registration state of the state machines associated with the MVRP application is achieved by means of the Registrar Administrative Control parameters provided by MRP (10.7.2). These parameters are represented as Static VLAN Registration Entries in the Filtering Database (8.8.2). Where management capability is implemented, these parameters can be manipulated by means of the management functionality defined in 12.7.

The provision of static control over the ability of Applicant state machines to participate in protocol exchanges is achieved by means of the Applicant Administrative Control parameters associated with the operation of MRP (10.7.3). Where management capability is implemented, the Applicant Administrative Control parameters can be applied and modified by means of the management functionality defined in 12.9.

Further administrative control over dynamic VLAN registration can be achieved, if supported, by means of a per-Port Restricted_VLAN_Registration control parameter. If the value of this control is TRUE for a given Port, the creation or modification of Dynamic VLAN Registration Entries as a result of MVRP exchanges on that Port shall be restricted only to those VIDs for which Static VLAN Registration Entries exist in which the Registrar Administrative Control value is Normal Registration. If the value of the Restricted_VLAN_Registration control is FALSE, dynamic VLAN registration is not so restricted. The recommended default value of this parameter is FALSE. Where management capability is implemented, the value of the Restricted_VLAN_Registration control can be manipulated by means of the management functionality defined in 12.10. If management of this parameter is not supported, the value of this parameter shall be FALSE for all Ports.

11.2.4 VID translation table

If a VID Translation Table (6.9) is in use for a Bridge Port, the VID values received in MVRP Attributes are translated on reception of MVRPDUs prior to MVRP processing as specified in this subclause (11.2), and translated after processing for encoding MVRP Attributes in transmitted MVRPDUs.

11.2.5 Use of “new” declaration capability

MVRP makes use of the “new” declaration capability in order to allow filtering database entries to be removed or rapidly aged out on a per-VID basis following a topology change in the network connectivity supporting the VID concerned. The rules for propagation of “new” declarations as defined in Clause 10 ensure that all MRP attribute declarations that originate from, or are propagated through, a bridge Port that has recently transitioned to Forwarding are marked as “new” declarations.

When any MVRP declaration marked as “new” is received on a given Port, either as a result of receiving an MVRPDU from the attached LAN (MAD_Join.indication), or as a result of receiving a request from MAP or the MVRP Application (MAD_Join.request), any entries in the filtering database for that Port and for the VID corresponding to the attribute value in the MAD_Join primitive are removed.

11.2.6 Attribute value support requirements

Implementations of MVRP shall be capable of supporting all attribute values in the range of possible values that can be registered using MVRP, and shall be capable of maintaining current state information for all attributes in the range of possible values.

12. Bridge management

This clause defines the set of managed objects, and their functionality, that allow administrative configuration of VLANs.

This clause

- a) Introduces the functions of management to assist in the identification of the requirements placed on Bridges for the support of management facilities.
- b) Establishes the correspondence between the Processes used to model the operation of the Bridge (8.3) and the managed objects of the Bridge.
- c) Specifies the management operations supported by each managed object.

12.1 Management functions

Management functions relate to the users' needs for facilities that support the planning, organization, supervision, control, protection, and security of communications resources, and account for their use. These facilities may be categorized as supporting the functional areas of Configuration, Fault, Performance, Security, and Accounting Management. Each functional area is summarized in 12.1.1 through 12.1.5, together with the facilities commonly required for the management of communication resources, and the particular facilities provided in that functional area by Bridge Management.

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

Configuration Management provides for the identification of communications resources, initialization, reset and close-down, the supply of operational parameters, and the establishment and discovery of the relationship between resources. The facilities provided by Bridge Management in this functional area are

- a) The identification of all Bridges that together make up the network and their respective locations and, as a consequence of that identification, the location of specific end stations to particular individual LANs.
- b) The ability to remotely reset, i.e., reinitialize, specified Bridges.
- c) The ability to control the priority with which a Bridge Port transmits frames.
- d) The ability to force a specific configuration of a spanning tree.
- e) The ability to control the propagation of frames with specific group MAC Addresses to certain parts of the configured network.
- f) The ability to identify the VLAN IDs (VIDs) in use, and through which Ports of the Bridge and for which Protocols frames destined for a given VID may be received and/or forwarded.
- g) The ability to create and delete the functional elements of Connectivity Fault Management and to control their operation.
- h) The ability to create and delete the functional elements of DDCFM and to control their operations.
- i) The ability to create and delete the functional elements of congestion notification and to control their operation.

12.1.2 Fault Management

Fault Management provides for fault prevention, detection, diagnosis, and correction. The facilities provided by Bridge Management in this functional area are

- a) The ability to identify and correct Bridge malfunctions, including error logging and reporting.
- b) 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.
 - 5) Determine whether a flow of specified data frames (e.g., frames associated with a service instance or selected data frames with certain destination address) can reach a particular destination.
 - 6) Determine whether a flow of data frames can be sent without error from a specific location within a network to a station or stations specified by the DA field of each frame.

12.1.3 Performance Management

Performance Management provides for evaluation of the behavior of communications resources and of the effectiveness of communication activities. The facilities provided by Bridge Management in this functional area are as follows:

- a) The ability to gather statistics relating to performance and traffic analysis. Specific metrics include network utilization, frame forward, and frame discard counts for individual Ports within a Bridge.

12.1.4 Security Management

Security Management provides for the protection of resources. Bridge Management does not provide any specific facilities in this functional area.

12.1.5 Accounting Management

Accounting Management provides for the identification and distribution of costs and the setting of charges. Bridge Management does not provide any specific facilities in this functional area.

12.2 VLAN-aware bridge objects

Managed objects model the semantics of management operations. Operations on an object supply information concerning, or facilitate control over, the Process or Entity associated with that object.

The managed resources of a VLAN-aware Bridge or bridge component are those of the Processes and Entities in 8.3. Specifically,

- a) The Bridge Management Entity (12.4 and 8.12).
- b) The individual MAC Entities associated with each Bridge Port (12.5, 8.2, and 8.5).
- c) The Forwarding Process of the MAC Relay Entity (12.6, 8.2, and 8.6).
- d) The Filtering Database of the MAC Relay Entity (12.7 and 8.8).
- e) The Bridge Protocol Entity (12.8 and 8.10).
- f) MRP participants (Clause 10);

- g) MVRP participants (12.10, Clause 11);
- h) MMRP participants (12.11, Clause 10);
- i) The MST Configuration Table (12.12).
- j) Additional objects to support Provider Bridge management (12.13);
- k) The CFM Entities (12.14).
- l) The DDCFM Entities (12.17).
- m) The PBB-TE protection switching Entities (12.18).
- n) The TPMR Entities (12.19).
- o) The management entities for forwarding and queuing for time sensitive streams (12.20).
- p) The congestion notification entities (12.21).

The management of each of these resources is described in terms of managed objects and operations in 12.4 through 12.12.

NOTE—The values specified in this clause, as inputs and outputs of management operations, are abstract information elements. Questions of formats or encodings are a matter for particular protocols that convey or otherwise represent this information.

Some data elements that are represented by the managed objects specified in this clause are required to be persistent across reinitializations or rebooting of the system within which the objects are implemented. For example, it may be desirable for changes in configuration parameters to persist across such events. Where such a requirement exists, this is indicated in the specification by marking the object concerned with the keyword “*Persistent*.”

12.3 Data types

This subclause specifies the semantics of operations independent of their encoding in management protocol. The data types of the parameters of operations are defined only as required for that specification.

The following data types are used:

- a) Boolean.
- b) Enumerated, for a collection of named values.
- c) Unsigned, for all parameters specified as “the number of” some quantity, and for Spanning Tree priority values that are numerically compared. When comparing Spanning Tree priority values, the lower number represents the higher priority value.
- d) MAC Address.
- e) Latin1 String, as defined by ANSI X3.159, for all text strings.
- f) Time Interval, an Unsigned value representing a positive integral number of seconds, for all Spanning Tree protocol timeout parameters;
- g) Counter, for all parameters specified as a “count” of some quantity. A counter increments and wraps with a modulus of 2 to the power of 64.
- h) MRP Time Interval, an Unsigned value representing a positive integral number of centiseconds, for all MRP timeout parameters.
- i) Port Number, an Unsigned value assigned to a Port as part of a Port Identifier. Valid Port Numbers are in the range 1 through 4095;
- j) Port Priority, an Unsigned value used to represent the priority component of a Port Identifier. Valid Port Priorities are in the range 0 through 240, in steps of 16;
- k) Bridge Priority, an Unsigned value used to represent the priority component of a Bridge Identifier. Valid Bridge Priorities are in the range 0 through 61440, in steps of 4096.
- l) ComponentID, an unsigned value used to uniquely identify the management objects for a particular VLAN-aware bridge component (12.2, 8, 5.4) within a system (such as a Backbone Edge Bridge) comprising multiple such components. ComponentIDs start at 1 and go through 4294967295. If the system has a single component it will have a ComponentID equal to 1;

- m) ComponentType, an enumerated list used to classify a particular VLAN-aware bridge component within a system comprising multiple components;
- n) Port Index, a handle, unique within a system, that identifies a port;
- o) PIP Index, a Port Index for a PIP;
- p) Percentage.

12.4 Bridge Management Entity

The Bridge Management Entity is described in 8.12.

The objects that comprise this managed resource are

- a) The Bridge Configuration (12.4.1).
- b) The Port Configuration for each Port (12.4.2).

12.4.1 Bridge Configuration

The Bridge Configuration object models the operations that modify, or inquire about, the configuration of the Bridge's resources. There is a single Bridge Configuration object per Bridge.

The management operations that can be performed on the Bridge Configuration are

- a) Discover Bridge (12.4.1.1);
- b) Read Bridge (12.4.1.2);
- c) Set Bridge Name (12.4.1.3);
- d) Reset Bridge (12.4.1.4).

12.4.1.1 Discover Bridge

12.4.1.1.1 Purpose

To solicit configuration information regarding the Bridge(s) in the network.

12.4.1.1.2 Inputs

- a) Inclusion Range, a set of ordered pairs of specific MAC Addresses. Each pair specifies a range of MAC Addresses. A Bridge shall respond if and only if
 - 1) For one of the pairs, the numerical comparison of its Bridge Address with each MAC Address of the pair shows it to be greater than or equal to the first, and
 - 2) Less than or equal to the second, and
 - 3) Its Bridge Address does not appear in the Exclusion List parameter below.

The numerical comparison of one MAC Address with another, for the purpose of this operation, is achieved by deriving a number from the MAC Address according to the following procedure. The consecutive octets of the MAC Address are taken to represent a binary number; the first octet that would be transmitted on a LAN medium when the MAC Address is used in the source or destination fields of a MAC frame has the most significant value, the next octet the next most significant value. Within each octet, the first bit of each octet is the LSB.

- b) Exclusion List, a list of specific MAC Addresses.

12.4.1.1.3 Outputs

- a) Bridge Address—the MAC Address for the Bridge from which the Bridge Identifiers used by the Spanning Tree Algorithm and Protocol, the Rapid Spanning Tree Protocol, and the Multiple Spanning Tree Protocol are derived (8.13.8, 13.24).
- b) Bridge Name—a text string of up to 32 characters, of locally determined significance.
- c) Number of Ports—the number of Bridge Ports (MAC Entities).
- d) Port Addresses—a list specifying the following for each Port:
 - 1) Port Number—the number of the Bridge Port (13.25).
 - 2) Port Address—the specific MAC Address of the individual MAC Entity associated with the Port (8.13.2).
- e) Uptime—count in seconds of the time elapsed since the Bridge was last reset or initialized.

NOTE—Events that are considered to reset or initialize the Bridge include changing the MST Configuration Identifier.

12.4.1.2 Read Bridge

12.4.1.2.1 Purpose

To obtain general information regarding the Bridge.

12.4.1.2.2 Inputs

None.

12.4.1.2.3 Outputs

- a) Bridge Address—the MAC Address for the Bridge from which the Bridge Identifiers used by the Spanning Tree Algorithm and Protocol and the Multiple Spanning Tree Protocol are derived (8.13.8, 13.24).
- b) Bridge Name—a text string of up to 32 characters, of locally determined significance.
- c) Number of Ports—the number of Bridge Ports (MAC Entities).
- d) Port Addresses—a list specifying the following for each Port:
 - 1) Port Number (13.25).
 - 2) Port Address—the specific MAC Address of the individual MAC Entity associated with the Port (8.13.2).
- e) Uptime—count in seconds of the time elapsed since the Bridge was last reset or initialized.

12.4.1.3 Set Bridge Name

12.4.1.3.1 Purpose

To associate a text string, readable by the Read Bridge operation, with a Bridge.

12.4.1.3.2 Inputs

- a) Bridge Name—a text string of up to 32 characters.

12.4.1.3.3 Outputs

None.

12.4.1.4 Reset Bridge

12.4.1.4.1 Purpose

To reset the specified Bridge. The Filtering Database is cleared and initialized with the entries specified in the Permanent Database, and the Bridge Protocol Entity is initialized.

12.4.1.4.2 Inputs

None.

12.4.1.4.3 Outputs

None.

12.4.2 Port configuration

The Port Configuration object models the operations that modify, or inquire about, the configuration of the Ports of a Bridge. There are a fixed set of Bridge Ports per Bridge (one for each MAC interface), and each is identified by a permanently allocated Port Number.

The allocated Port Numbers are not required to be consecutive. Also, some Port Numbers may be dummy entries, with no actual LAN Port (for example, to allow for expansion of the Bridge by addition of further MAC interfaces in the future). Such dummy Ports shall support the Port Configuration management operations and other Port-related management operations in a manner consistent with the Port being permanently disabled.

The information provided by the Port Configuration consists of summary data indicating its name and type. Specific counter information pertaining to the number of packets forwarded, filtered, and in error is maintained by the Forwarding Process resource. The management operations supported by the Bridge Protocol Entity allow for controlling the states of each Port.

The management operations that can be performed on the Port Configuration are

- a) Read Port (12.4.2.1);
- b) Set Port Name (12.4.2.2).

12.4.2.1 Read Port

12.4.2.1.1 Purpose

To obtain general information regarding a specific Bridge Port.

12.4.2.1.2 Inputs

- a) Port Number—the number of the Bridge Port (13.25).

12.4.2.1.3 Outputs

- a) Port Name—a text string of up to 32 characters, of locally determined significance.
- b) Port Type—the MAC Entity type of the Port (IEEE Std 802.3; ISO/IEC 8802-4; ISO/IEC 8802-5; ISO/IEC 8802-6; ISO/IEC 8802-9; IEEE Std 802.9a-1995; ISO/IEC 8802-11; ISO/IEC 8802-12 (IEEE 802.3 format); ISO/IEC 8802-12 (ISO/IEC 8802-5 format); ISO 9314; other).

12.4.2.2 Set Port Name

12.4.2.2.1 Purpose

To associate a text string, readable by the Read Port operation, with a Bridge Port.

12.4.2.2.2 Inputs

- a) Port Number (13.25).
- b) Port Name—a text string of up to 32 characters.

12.4.2.2.3 Outputs

None.

12.5 MAC entities

The Management Operations and Facilities provided by the MAC Entities are those specified in the Layer Management standards of the individual MACs. A MAC Entity is associated with each Bridge Port.

12.6 Forwarding process

The Forwarding Process contains information relating to the forwarding of frames. Counters are maintained that provide information on the number of frames forwarded, filtered, and dropped due to error. Configuration data, defining how frame priority is handled, is maintained by the Forwarding Process.

The objects that comprise this managed resource are

- a) The Port Counters (12.6.1).
- b) The Priority Handling objects for each Port (12.6.2).
- c) The Traffic Class Table for each Port (12.6.3).

12.6.1 The Port Counters

The Port Counters object models the operations that can be performed on the Port counters of the Forwarding Process resource. There are multiple instances (one for each VID for each MAC Entity) of the Port Counters object per Bridge.

The management operation that can be performed on the Port Counters is Read Forwarding Port Counters (12.6.1.1).

12.6.1.1 Read forwarding port counters

12.6.1.1.1 Purpose

To read the forwarding counters associated with a specific Bridge Port.

12.6.1.1.2 Inputs

- a) Port Number (13.25);
- b) Optionally, VLAN Identifier (9.6).

If the VLAN Identifier parameter is supported, then the forwarding Port counters are maintained per VID per Port. If the parameter is not supported, then the forwarding Port counters are maintained per Port only.

12.6.1.1.3 Outputs

- a) Frames Received—count of all valid frames received (including BPDUs, frames addressed to the Bridge as an end station, and frames that were submitted to the Forwarding Process, 8.5).
- b) Optionally, Octets Received—count of the total number of octets in all valid frames received (including BPDUs, frames addressed to the Bridge as an end station, and frames that were submitted to the Forwarding Process).
- c) Discard Inbound—count of valid frames received that were discarded by the Forwarding Process (8.6).
- d) Forward Outbound—count of frames forwarded to the associated MAC Entity (8.5).
- e) Discard Lack of Buffers—count of frames that were to be transmitted through the associated Port but were discarded due to lack of buffers (8.6.6).
- f) Discard Transit Delay Exceeded—count of frames that were to be transmitted but were discarded due to the maximum bridge transit delay being exceeded (buffering may have been available, 8.6.6).
- g) Discard on Error—count of frames that were to be forwarded on the associated MAC but could not be transmitted (e.g., frame would be too large, 6.5.8).
- h) If Ingress Filtering is supported (8.6.2), Discard on Ingress Filtering—count of frames that were discarded as a result of Ingress Filtering being enabled.

12.6.2 Priority handling

The Priority Handling object models the operations that can be performed on, or inquire about, the Default Priority parameter, the Priority Regeneration Table parameter, the Outbound Access Priority Table parameter, the Priority Code Point parameters, and the Service Access Priority parameters for each Port. The operations that can be performed on this object are

- a) Read Port Default Priority (12.6.2.1);
- b) Set Port Default Priority (12.6.2.2);
- c) Read Port Priority Regeneration Table (12.6.2.3);
- d) Set Port Priority Regeneration Table (12.6.2.4);
- e) Read Port Priority Code Point Selection (12.6.2.5);
- f) Set Port Priority Code Point Selection (12.6.2.6);
- g) Read Port Priority Code Point Decoding Table (12.6.2.7);
- h) Optionally, Set Port Priority Code Point Decoding Table (12.6.2.8);
- i) Read Port Priority Code Point Encoding Table (12.6.2.9);
- j) Optionally, Set Port Priority Code Point Encoding Table (12.6.2.10);
- k) Read Use_DEI parameter (12.6.2.12);
- l) Optionally, Set Use_DEI parameter (12.6.2.13);
- m) Read Require Drop Encoding parameter (12.6.2.14);
- n) Optionally, Set Require Drop Encoding parameter (12.6.2.15);
- o) Read Service Access Priority Selection (12.6.2.16);
- p) Optionally, Set Service Access Priority Selection (12.6.2.17);
- q) Read Service Access Priority Table (12.6.2.18);
- r) Optionally, Set Service Access Priority Table (12.6.2.19).

12.6.2.1 Read Port Default Priority

12.6.2.1.1 Purpose

To read the current state of the Default Priority parameter (6.6) for a specific Bridge Port.

12.6.2.1.2 Inputs

- a) Port number.

12.6.2.1.3 Outputs

- a) Default Priority value—Integer in range 0–7.

12.6.2.2 Set Port Default Priority**12.6.2.2.1 Purpose**

To set the current state of the Default Priority parameter (6.6) for a specific Bridge Port.

12.6.2.2.2 Inputs

- a) Port number;
- b) Default Priority value—Integer in range 0–7.

12.6.2.2.3 Outputs

None.

12.6.2.3 Read Port Priority Regeneration Table**12.6.2.3.1 Purpose**

To read the current state of the Priority Regeneration Table parameter (6.9.4) for a specific Bridge Port.

12.6.2.3.2 Inputs

- a) Port number.

12.6.2.3.3 Outputs

- a) Regenerated Priority value for Received Priority 0—Integer in range 0–7.
- b) Regenerated Priority value for Received Priority 1—Integer in range 0–7.
- c) Regenerated Priority value for Received Priority 2—Integer in range 0–7.
- d) Regenerated Priority value for Received Priority 3—Integer in range 0–7.
- e) Regenerated Priority value for Received Priority 4—Integer in range 0–7.
- f) Regenerated Priority value for Received Priority 5—Integer in range 0–7.
- g) Regenerated Priority value for Received Priority 6—Integer in range 0–7.
- h) Regenerated Priority value for Received Priority 7—Integer in range 0–7.

12.6.2.4 Set Port Priority Regeneration Table**12.6.2.4.1 Purpose**

To set the current state of the Priority Regeneration Table parameter (6.9.4) for a specific Bridge Port.

12.6.2.4.2 Inputs

- a) Port number;
- b) Regenerated Priority value for Received Priority 0—Integer in range 0–7.

- c) Regenerated Priority value for Received Priority 1—Integer in range 0–7.
- d) Regenerated Priority value for Received Priority 2—Integer in range 0–7.
- e) Regenerated Priority value for Received Priority 3—Integer in range 0–7.
- f) Regenerated Priority value for Received Priority 4—Integer in range 0–7.
- g) Regenerated Priority value for Received Priority 5—Integer in range 0–7.
- h) Regenerated Priority value for Received Priority 6—Integer in range 0–7.
- i) Regenerated Priority value for Received Priority 7—Integer in range 0–7.

12.6.2.4.3 Outputs

None.

12.6.2.5 Read Port Priority Code Point Selection

12.6.2.5.1 Purpose

To read which row of the Priority Code Point Encoding Table and Priority Code Point Decoding Table (6.9.3) is currently selected for use on this Port.

12.6.2.5.2 Inputs

- a) Port Number: the number of the Bridge Port.

12.6.2.5.3 Outputs

- a) Priority Code Point Selection: this takes one of the following values:
 - 1) 8P0D;
 - 2) 7P1D;
 - 3) 6P2D;
 - 4) 5P3D.

12.6.2.6 Set Port Priority Code Point Selection

12.6.2.6.1 Purpose

To set which row of the Priority Code Point Encoding Table and Priority Code Point Decoding Table (6.9.3) will be selected for use on this Port.

12.6.2.6.2 Inputs

- a) Port Number: the number of the Bridge Port.
- b) Priority Code Point Selection: this takes one of the following values:
 - 1) 8P0D;
 - 2) 7P1D;
 - 3) 6P2D;
 - 4) 5P3D.

12.6.2.6.3 Outputs

None.

12.6.2.7 Read Priority Code Point Decoding Table

12.6.2.7.1 Purpose

To read the current contents of a row in the Priority Code Point Decoding Table (6.9.3) for a Port.

12.6.2.7.2 Inputs

- a) Port Number: the number of the Bridge Port.
- b) Priority Code Point Row: this takes one of the following values:
 - 1) 8P0D;
 - 2) 7P1D;
 - 3) 6P2D;
 - 4) 5P3D.

12.6.2.7.3 Outputs

- a) Priority value for Priority Code Point 0: Integer in range 0–7;
- b) Drop_eligible value for Priority Code Point 0: Boolean;
- c) Priority value for Priority Code Point 1: Integer in range 0–7;
- d) Drop_eligible value for Priority Code Point 1: Boolean;
- e) Priority value for Priority Code Point 2: Integer in range 0–7;
- f) Drop_eligible value for Priority Code Point 2: Boolean;
- g) Priority value for Priority Code Point 3: Integer in range 0–7;
- h) Drop_eligible value for Priority Code Point 3: Boolean;
- i) Priority value for Priority Code Point 4: Integer in range 0–7;
- j) Drop_eligible value for Priority Code Point 4: Boolean;
- k) Priority value for Priority Code Point 5: Integer in range 0–7;
- l) Drop_eligible value for Priority Code Point 5: Boolean;
- m) Priority value for Priority Code Point 6: Integer in range 0–7;
- n) Drop_eligible value for Priority Code Point 6: Boolean;
- o) Priority value for Priority Code Point 7: Integer in range 0–7;
- p) Drop_eligible value for Priority Code Point 7: Boolean.

12.6.2.8 Set Priority Code Point Decoding Table

12.6.2.8.1 Purpose

To modify the contents of a row in the Priority Code Point Decoding Table (6.9.3) for a Port.

12.6.2.8.2 Inputs

- a) Port Number: the number of the Bridge Port;
- b) Priority Code Point Row: this takes one of the following values:
 - 1) 8P0D;
 - 2) 7P1D;
 - 3) 6P2D;
 - 4) 5P3D;
- c) Priority value for Priority Code Point 0: Integer in range 0–7;
- d) Drop_eligible value for Priority Code Point 0: Boolean;
- e) Priority value for Priority Code Point 1: Integer in range 0–7;
- f) Drop_eligible value for Priority Code Point 1: Boolean;
- g) Priority value for Priority Code Point 2: Integer in range 0–7;
- h) Drop_eligible value for Priority Code Point 2: Boolean;

- i) Priority value for Priority Code Point 3: Integer in range 0–7;
- j) Drop_eligible value for Priority Code Point 3: Boolean;
- k) Priority value for Priority Code Point 4: Integer in range 0–7;
- l) Drop_eligible value for Priority Code Point 4: Boolean;
- m) Priority value for Priority Code Point 5: Integer in range 0–7;
- n) Drop_eligible value for Priority Code Point 5: Boolean;
- o) Priority value for Priority Code Point 6: Integer in range 0–7;
- p) Drop_eligible value for Priority Code Point 6: Boolean;
- q) Priority value for Priority Code Point 7: Integer in range 0–7;
- r) Drop_eligible value for Priority Code Point 7: Boolean.

12.6.2.8.3 Outputs

None.

12.6.2.9 Read Priority Code Point Encoding Table

12.6.2.9.1 Purpose

To read the current contents of a row in the Priority Code Point Encoding Table (6.9.3) for a Port.

12.6.2.9.2 Inputs

- a) Port Number: the number of the Bridge Port;
- b) Priority Code Point Row: this takes one of the following values:
 - 1) 8P0D;
 - 2) 7P1D;
 - 3) 6P2D;
 - 4) 5P3D.

12.6.2.9.3 Outputs

- a) Priority Code Point value for priority 0 with drop_eligible False: Integer in range 0–7;
- b) Priority Code Point value for priority 0 with drop_eligible True: Integer in range 0–7;
- c) Priority Code Point value for priority 1 with drop_eligible False: Integer in range 0–7;
- d) Priority Code Point value for priority 1 with drop_eligible True: Integer in range 0–7;
- e) Priority Code Point value for priority 2 with drop_eligible False: Integer in range 0–7;
- f) Priority Code Point value for priority 2 with drop_eligible True: Integer in range 0–7;
- g) Priority Code Point value for priority 3 with drop_eligible False: Integer in range 0–7;
- h) Priority Code Point value for priority 3 with drop_eligible True: Integer in range 0–7;
- i) Priority Code Point value for priority 4 with drop_eligible False: Integer in range 0–7;
- j) Priority Code Point value for priority 4 with drop_eligible True: Integer in range 0–7;
- k) Priority Code Point value for priority 5 with drop_eligible False: Integer in range 0–7;
- l) Priority Code Point value for priority 5 with drop_eligible True: Integer in range 0–7;
- m) Priority Code Point value for priority 6 with drop_eligible False: Integer in range 0–7;
- n) Priority Code Point value for priority 6 with drop_eligible True: Integer in range 0–7;
- o) Priority Code Point value for priority 7 with drop_eligible False: Integer in range 0–7;
- p) Priority Code Point value for priority 7 with drop_eligible True: Integer in range 0–7.

12.6.2.10 Set Priority Code Point Encoding Table

12.6.2.10.1 Purpose

To modify the contents of a row in the Priority Code Point Encoding Table (6.9.3) for a Port.

12.6.2.10.2 Inputs

- a) Port Number: the number of the Bridge Port;
- b) Priority Code Point Row: this takes one of the following values:
 - 1) 8P0D;
 - 2) 7P1D;
 - 3) 6P2D;
 - 4) 5P3D;
- c) Priority Code Point value for priority 0 with drop_eligible False: Integer in range 0–7;
- d) Priority Code Point value for priority 0 with drop_eligible True: Integer in range 0–7;
- e) Priority Code Point value for priority 1 with drop_eligible False: Integer in range 0–7;
- f) Priority Code Point value for priority 1 with drop_eligible True: Integer in range 0–7;
- g) Priority Code Point value for priority 2 with drop_eligible False: Integer in range 0–7;
- h) Priority Code Point value for priority 2 with drop_eligible True: Integer in range 0–7;
- i) Priority Code Point value for priority 3 with drop_eligible False: Integer in range 0–7;
- j) Priority Code Point value for priority 3 with drop_eligible True: Integer in range 0–7;
- k) Priority Code Point value for priority 4 with drop_eligible False: Integer in range 0–7;
- l) Priority Code Point value for priority 4 with drop_eligible True: Integer in range 0–7;
- m) Priority Code Point value for priority 5 with drop_eligible False: Integer in range 0–7;
- n) Priority Code Point value for priority 5 with drop_eligible True: Integer in range 0–7;
- o) Priority Code Point value for priority 6 with drop_eligible False: Integer in range 0–7;
- p) Priority Code Point value for priority 6 with drop_eligible True: Integer in range 0–7;
- q) Priority Code Point value for priority 7 with drop_eligible False: Integer in range 0–7;
- r) Priority Code Point value for priority 7 with drop_eligible True: Integer in range 0–7.

12.6.2.10.3 Outputs

None.

12.6.2.11 Read Use_DEI Parameter**12.6.2.12.1 Purpose**

To read the current state of the Use_DEI parameter (6.9.3) for the Port.

12.6.2.12.2 Inputs

- a) Port Number: the number of the Bridge Port.

12.6.2.12.3 Outputs

- a) Use_DEI parameter: Boolean.

12.6.2.13 Set Use_DEI Parameter**12.6.2.13.1 Purpose**

To set the current state of the Use_DEI parameter (6.9.3) for the Port.

12.6.2.13.2 Inputs

- a) Port Number: the number of the Bridge Port.
- a) Use_DEI parameter: Boolean.

12.6.2.13.3 Outputs

None.

12.6.2.14 Read Require Drop Encoding Parameter

12.6.2.14.1 Purpose

To read the current state of the Require Drop Encoding parameter (8.6.6) for the Port.

12.6.2.14.2 Inputs

- a) Port Number: the number of the Bridge Port.

12.6.2.14.3 Outputs

- a) Require Drop Encoding parameter: Boolean.

12.6.2.15 Set Require Drop Encoding Parameter

12.6.2.15.1 Purpose

To set the current state of the Require Drop Encoding parameter (8.6.6) for the Port.

12.6.2.15.2 Inputs

- a) Port Number: the number of the Bridge Port.
- b) Require Drop Encoding parameter: Boolean.

12.6.2.15.3 Outputs

None.

12.6.2.16 Read Service Access Priority Selection

12.6.2.16.1 Purpose

To read the current state of whether Service Access Priority Selection is enabled (6.13) for the Port.

12.6.2.16.2 Inputs

- a) Port Number: the number of the Bridge Port.

12.6.2.16.3 Outputs

- a) Enable Service Access Priority Selection: the permissible values are as follows:
 - 1) Enabled;
 - 2) Disabled.

12.6.2.17 Set Service Access Priority Selection

12.6.2.17.1 Purpose

To enable or disable Service Access Priority Selection (6.13) for the Port.

12.6.2.17.2 Inputs

- a) Port Number: the number of the Bridge Port.
- b) Service Access Priority Selection: the permissible values are as follows:
 - 1) Enabled;
 - 2) Disabled.

12.6.2.17.3 Outputs

None.

12.6.2.18 Read Service Access Priority Table

12.6.2.18.1 Purpose

To read the current contents of the Service Access Priority Table (6.13.1) for the Port.

12.6.2.18.2 Inputs

- a) Port Number: the number of the Bridge Port.

12.6.2.18.3 Outputs

- a) Service Access Priority value for Received Priority 0: Integer in range 0–7;
- b) Service Access Priority value for Received Priority 1: Integer in range 0–7;
- c) Service Access Priority value for Received Priority 2: Integer in range 0–7;
- d) Service Access Priority value for Received Priority 3: Integer in range 0–7;
- e) Service Access Priority value for Received Priority 4: Integer in range 0–7;
- f) Service Access Priority value for Received Priority 5: Integer in range 0–7;
- g) Service Access Priority value for Received Priority 6: Integer in range 0–7;
- h) Service Access Priority value for Received Priority 7: Integer in range 0–7.

12.6.2.19 Set Service Access Priority Table

12.6.2.19.1 Purpose

To modify the contents of the Service Access Priority Table (6.13.1) for the Port.

12.6.2.19.2 Inputs

- a) Port Number: the number of the Bridge Port;
- b) Service Access Priority value for Received Priority 0: Integer in range 0–7;
- c) Service Access Priority value for Received Priority 1: Integer in range 0–7;
- d) Service Access Priority value for Received Priority 2: Integer in range 0–7;
- e) Service Access Priority value for Received Priority 3: Integer in range 0–7;
- f) Service Access Priority value for Received Priority 4: Integer in range 0–7;
- g) Service Access Priority value for Received Priority 5: Integer in range 0–7;
- h) Service Access Priority value for Received Priority 6: Integer in range 0–7;
- i) Service Access Priority value for Received Priority 7: Integer in range 0–7.

12.6.2.19.3 Outputs

None.

12.6.3 Traffic Class Table

The Traffic Class Table object models the operations that can be performed on, or inquire about, the current contents of the Traffic Class Table (8.6.6) for a given Port. The operations that can be performed on this object are Read Port Traffic Class Table and Set Port Traffic Class Table.

12.6.3.1 Read Port Traffic Class Table

12.6.3.1.1 Purpose

To read the contents of the Traffic Class Table (8.6.6) for a given Port.

12.6.3.1.2 Inputs

- a) Port Number.

12.6.3.1.3 Outputs

- a) The number of traffic classes, in the range 1 through 8, supported on the Port;
- b) For each value of traffic class supported on the Port, the value of the traffic class in the range 0 through 7, and the set of priority values assigned to that traffic class.

12.6.3.2 Set Port Traffic Class Table

12.6.3.2.1 Purpose

To set the contents of the Traffic Class Table (8.6.6) for a given Port.

12.6.3.2.2 Inputs

- a) Port number;
- b) For each value of traffic class supported on the Port, the value of the traffic class in the range 0 through 7, and the set of priority values assigned to that traffic class.

NOTE—If a traffic class value greater than the largest traffic class available on the Port is specified, then the value applied to the Traffic Class Table is the largest available traffic class.

12.6.3.2.3 Outputs

None.

12.7 Filtering Database

The Filtering Database is described in 8.8. It contains filtering information used by the Forwarding Process (8.6) in deciding through which Ports of the Bridge frames should be forwarded.

The objects that comprise this managed resource are

- a) The Filtering Database (12.7.1);
- b) The Static Filtering Entries (12.7.2);
- c) The Dynamic Filtering Entries (12.7.3);
- d) The MAC Address Registration Entries (12.7.4);
- e) The Static VLAN Registration Entries (12.7.5);

- f) The Dynamic VLAN Registration Entries (12.7.5);
- g) The Permanent Database (12.7.6).

12.7.1 The Filtering Database

The Filtering Database object models the operations that can be performed on, or affect, the Filtering Database as a whole. There is a single Filtering Database object per Bridge.

The management operations that can be performed on the Database are:

- a) Read Filtering Database (12.7.1.1);
- b) Set Filtering Database Ageing Time (12.7.1.2);
- c) Read Permanent Database (12.7.6.1);
- d) Create Filtering Entry (12.7.7.1);
- e) Delete Filtering Entry (12.7.7.2);
- f) Read Filtering Entry (12.7.7.3);
- g) Read Filtering Entry Range (12.7.7.4).

12.7.1.1 Read Filtering Database

12.7.1.1.1 Purpose

To obtain general information regarding the Bridge's Filtering Database.

12.7.1.1.2 Inputs

None.

12.7.1.1.3 Outputs

- a) Filtering Database Size—the maximum number of entries that can be held in the Filtering Database;
- b) Number of Static Filtering Entries—the number of Static Filtering Entries (8.8.1) currently in the Filtering Database;
- c) Number of Dynamic Filtering Entries—the number of Dynamic Filtering Entries (8.8.3) currently in the Filtering Database;
- d) Number of Static VLAN Registration Entries—the number of Static VLAN Registration Entries (8.8.2) currently in the Filtering Database;
- e) Number of Dynamic VLAN Registration Entries—the number of Dynamic VLAN Registration Entries (8.8.5) currently in the Filtering Database;
- f) Ageing Time—for ageing out Dynamic Filtering Entries when the Port associated with the entry is in the Forwarding state (8.8.3);
- g) If Extended Filtering Services are supported, Number of MAC Address Registration Entries—the number of MAC Address Registration Entries (8.8.4) currently in the Filtering Database.

12.7.1.2 Set Filtering Database Ageing Time

12.7.1.2.1 Purpose

To set the ageing time for Dynamic Filtering Entries (8.8.3).

12.7.1.2.2 Inputs

- a) Ageing Time.

12.7.1.2.3 Outputs

None.

12.7.2 A Static Filtering Entry

A Static Filtering Entry object models the operations that can be performed on a single Static Filtering Entry in the Filtering Database. The set of Static Filtering Entry objects within the Filtering Database changes only under management control.

A Static Filtering Entry object supports the following operations:

- a) Create Filtering Entry (12.7.7.1);
- b) Delete Filtering Entry (12.7.7.2);
- c) Read Filtering Entry (12.7.7.3);
- d) Read Filtering Entry Range (12.7.7.4).

12.7.3 A Dynamic Filtering Entry

A Dynamic Filtering Entry object models the operations that can be performed on a single Dynamic Filtering Entry (i.e., one that is created by the Learning Process as a result of the observation of network traffic) in the Filtering Database.

A Dynamic Filtering Entry object supports the following operations:

- a) Delete Filtering Entry (12.7.7.2);
- b) Read Filtering Entry (12.7.7.3);
- c) Read Filtering Entry Range (12.7.7.4).

12.7.4 A MAC Address Registration Entry

A MAC Address Registration Entry object models the operations that can be performed on a single MAC Address Registration Entry in the Filtering Database. The set of MAC Address Registration Entry objects within the Filtering Database changes only as a result of MMRP exchanges.

A MAC Address Registration Entry object supports the following operations:

- a) Read Filtering Entry (12.7.7.3);
- b) Read Filtering Entry Range (12.7.7.4).

12.7.5 A VLAN Registration Entry

A VLAN Registration Entry object models the operations that can be performed on a single VLAN Registration Entry in the Filtering Database. The set of VLAN Registration Entry objects within the Filtering Database changes under management control and also as a result of MVRP exchanges.

12.7.5.1 Static VLAN Registration Entry object

A Static VLAN Registration Entry object supports the following operations:

- a) Create Filtering Entry (12.7.7.1);
- b) Delete Filtering Entry (12.7.7.2);
- c) Read Filtering Entry (12.7.7.3);
- d) Read Filtering Entry Range (12.7.7.4).

12.7.5.2 Dynamic VLAN Registration Entry object

A Dynamic VLAN Registration Entry object supports the following operations:

- a) Read Filtering Entry (12.7.7.3);
- b) Read Filtering Entry Range (12.7.7.4).

12.7.6 Permanent Database

The Permanent Database object models the operations that can be performed on, or affect, the Permanent Database. There is a single Permanent Database per Filtering Database.

The management operations that can be performed on the Permanent Database are

- a) Read Permanent Database (12.7.6.1);
- b) Create Filtering Entry (12.7.7.1);
- c) Delete Filtering Entry (12.7.7.2);
- d) Read Filtering Entry (12.7.7.3);
- e) Read Filtering Entry Range (12.7.7.4).

12.7.6.1 Read Permanent Database

12.7.6.1.1 Purpose

To obtain general information regarding the Permanent Database (8.8.11).

12.7.6.1.2 Inputs

None.

12.7.6.1.3 Outputs

- a) Permanent Database Size—maximum number of entries that can be held in the Permanent Database;
- b) Number of Static Filtering Entries—number of Static Filtering Entries (8.8.1) currently in the Permanent Database;
- c) Number of Static VLAN Registration Entries—number of Static VLAN Registration Entries (8.8.2) currently in the Permanent Database.

12.7.7 General Filtering Database operations

In these operations on the Filtering Database, the operation parameters make use of VID values, even when operating on a Dynamic Filtering Entry (8.8.3) whose structure carries an FID rather than a VID. In this case, the value used in the VID parameter can be any VID that has been allocated to the FID concerned (8.8.8).

12.7.7.1 Create Filtering Entry

12.7.7.1.1 Purpose

To create or update a Static Filtering Entry (8.8.1) or Static VLAN Registration Entry (8.8.2) in the Filtering Database or Permanent Database. Only static entries may be created in the Filtering Database or Permanent Database.

12.7.7.1.2 Inputs

- a) Identifier—Filtering Database or Permanent Database.
- b) Address—MAC Address of the entry (not present in VLAN Registration Entries).
- c) VID—VLAN Identifier of the entry.
- d) Port Map—a set of control indicators, one for each Port, as specified in 8.8.1 and 8.8.2.

12.7.7.1.3 Outputs

- a) Operation rejected because a Port identified by the Port Map includes a port already in the member set of a VID of a different type than the currently registered VID.
- b) Operation accepted.

12.7.7.2 Delete Filtering Entry

12.7.7.2.1 Purpose

To delete a Filtering Entry or VLAN Registration Entry from the Filtering Database or Permanent Database.

12.7.7.2.2 Inputs

- a) Identifier—Filtering Database or Permanent Database.
- b) Address—MAC Address of the desired entry (not present in VLAN Registration Entries).
- c) VID—VLAN Identifier of the entry.

12.7.7.2.3 Outputs

None.

12.7.7.3 Read Filtering Entry

12.7.7.3.1 Purpose

To read a Filtering Entry, MAC Address Registration Entry, or VLAN Registration Entry from the Filtering or Permanent Databases.

12.7.7.3.2 Inputs

- a) Identifier—Filtering Database or Permanent Database.
- b) Address—MAC Address of the desired entry (not present in VLAN Registration Entries).
- c) VID—VLAN Identifier of the entry.
- d) Type—Static or Dynamic entry.

12.7.7.3.3 Outputs

- a) Address—MAC Address of the desired entry (not present in VLAN Registration Entries).
- b) VID—VLAN Identifier of the entry.
- c) Type—Static or Dynamic entry.
- d) Port Map—a set of control indicators as appropriate for the entry (that may include a Connection Identifier), as specified in 8.8.1 through 8.8.5.

12.7.7.4 Read Filtering Entry range

12.7.7.4.1 Purpose

To read a range of Filtering Database entries (of any type) from the Filtering or Permanent Databases.

Since the number of values to be returned in the requested range may have exceeded the capacity of the service data unit conveying the management response, the returned entry range is identified. The indices that define the range take on values from zero up to Filtering Database Size minus one.

12.7.7.4.2 Inputs

- a) Identifier—Filtering Database or Permanent Database.
- b) Start Index—inclusive starting index of the desired entry range.
- c) Stop Index—inclusive ending index of the desired range.

12.7.7.4.3 Outputs

- a) Start Index—inclusive starting index of the returned entry range.
- b) Stop Index—inclusive ending index of the returned entry range.
- c) For each index returned:
 - 1) Address—MAC Address of the desired entry (not present in VLAN Registration Entries).
 - 2) VID—VLAN Identifier of the entry.
 - 3) Type—Static or Dynamic entry.
 - 4) Port Map—a set of control indicators as appropriate for the entry (that may include a Connection Identifier), as specified in 8.8.1 through 8.8.5.

12.8 Bridge Protocol Entity

The Bridge Protocol Entity is described in 8.10 and Clause 13.

The objects that comprise this managed resource are

- a) The Protocol Entity.
- b) The Ports under its control.

12.8.1 The Protocol Entity

The Protocol Entity object models the operations that can be performed on, or inquire about, the operation of the Spanning Tree Algorithm and Protocol. There is a single Protocol Entity per Bridge; it can, therefore, be identified as a single fixed component of the Protocol Entity resource.

The management operations that can be performed on the Protocol Entity are

- a) Read CIST Bridge Protocol Parameters (12.8.1.1);
- b) Read MSTI Bridge Protocol Parameters (12.8.1.2);
- c) Set CIST Bridge Protocol Parameters (12.8.1.3);
- d) Set MSTI Bridge Protocol Parameters (12.8.1.4).

12.8.1.1 Read CIST Bridge Protocol Parameters

12.8.1.1.1 Purpose

To obtain information regarding the Bridge's Bridge Protocol Entity for the CIST.

12.8.1.1.2 Inputs

None.

12.8.1.1.3 Outputs

- a) Bridge Identifier—as defined in 13.24.1. The Bridge Identifier for the CIST.
- b) Time Since Topology Change—in an STP Bridge, the count in seconds of the time elapsed since the Topology Change flag parameter for the Bridge (8.5.3.12 of IEEE Std 802.1D, 1998 Edition) was last True, or in an RSTP or MSTP Bridge, the count in seconds since tcWhile timer (13.23) for any Port was nonzero.
- c) Topology Change Count—in an STP Bridge, the count of the times the Topology Change flag parameter for the Bridge has been set (i.e., transitioned from False to True) since the Bridge was powered on or initialized, or in an RSTP or MSTP Bridge, the count of times that there has been at least one nonzero tcWhile timer (13.23).
- d) CIST Root Identifier (13.24.8).
- e) CIST External Root Path Cost (13.24.8).
- f) Root Port (13.24.7).
- g) Max Age (13.24.9).
- h) Forward Delay (13.24.9).
- i) Bridge Max Age (13.24.3).
- j) Bridge Hello Time (13.23). This parameter is present only if the Bridge supports STP or RSTP.
- k) Bridge Forward Delay (13.24.3).
- l) Hold Time (8.5.3.14 of IEEE Std 802.1D, 1998 Edition) or Transmission Limit (TxHoldCount in 13.24.10).

The following parameter is present only if the Bridge supports RSTP or MSTP:

- m) forceVersion—the value of the Force Protocol Version parameter for the Bridge (13.6.2)

The following additional parameters are present only if the Bridge supports MSTP:

- n) CIST Regional Root Identifier (13.24.8). The Bridge Identifier of the current CIST Regional Root.
- o) CIST Path Cost (CIST Internal Root Path Cost, in 13.24.8). The CIST path cost from the transmitting Bridge to the CIST Regional Root.
- p) MaxHops (13.24.3).

12.8.1.2 Read MSTI Bridge Protocol Parameters

12.8.1.2.1 Purpose

In an MST Bridge, to obtain information regarding the Bridge's Bridge Protocol Entity for the specified Spanning Tree instance.

12.8.1.2.2 Inputs

- a) MSTID—Identifies the set of parameters that will be returned. For Bridges that support MSTP, this parameter is the identifier of the spanning tree for which the operation is being performed. This parameter takes a value in the range 1 through 4094.

12.8.1.2.3 Outputs

- a) MSTID—identifies the set of parameters that are being returned. This parameter is the identifier of the MST Instance for which the operation is being performed.
- b) Bridge Identifier—as defined in 13.24.1. The Bridge Identifier for the spanning tree instance identified by the MSTID.
- c) Time Since Topology Change—count in seconds of the time elapsed since tcWhile (13.23) was last nonzero for any Port for the given MSTI.
- d) Topology Change Count—count of the times tcWhile (13.23) has been nonzero for any Port for the given MSTI since the Bridge was powered on or initialized.
- e) Topology Change (tcWhile, 13.23). True if tcWhile is nonzero for any Port for the given MST.
- f) Designated Root (MSTI Regional Root Identifier, 13.24.8). The Bridge Identifier of the Root Bridge for the spanning tree instance identified by the MSTID.
- g) Root Path Cost (MSTI Internal Root Path Cost, 13.24.8). The path cost from the transmitting Bridge to the Root Bridge for the spanning tree instance identified by the MSTID.
- h) Root Port (13.24.7). The Root Port for the spanning tree instance identified by the MSTID.

12.8.1.3 Set CIST Bridge Protocol Parameters

12.8.1.3.1 Purpose

To modify parameters in the Bridge’s Bridge Protocol Entity for the CIST, in order to force a configuration of the spanning tree and/or tune the reconfiguration time to suit a specific topology. In RSTP and MSTP implementations, this operation causes these values to be set for all Ports of the Bridge.

12.8.1.3.2 Inputs

- a) Bridge Max Age—the new value (13.24.3).
- b) Bridge Hello Time—the new value (13.23). This parameter is present only if the Bridge supports STP or RSTP.
- c) Bridge Forward Delay—the new value (13.24.3).
- d) Bridge Priority—the new value of the priority part of the Bridge Identifier (13.24.1) for the CIST.

The following parameters are present only if the Bridge supports RSTP or MSTP:

- e) forceVersion—the new value of the Force Protocol Version parameter for the Bridge (13.6.2).
- f) TxHoldCount—the new value of TxHoldCount (13.24.10).

The following parameter is present only if the Bridge supports MSTP:

- g) MaxHops—the new value of MaxHops (13.24.3).

12.8.1.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to invalid Bridge Priority value (12.3); or
 - 2) Operation rejected due to the specified Max Age, Hello Time, or Forward Delay values being outside the range specified for the parameter (see 12.8.1.3.4); or

- 3) Operation rejected due to the specified Max Age, Hello Time, or Forward Delay values not being in compliance with the requirements of this standard (see 12.8.1.3.4); or
- 4) Operation rejected due to the specified MaxHops value not being within the permitted range specified in Table 13-5.
- 5) Operation accepted.

12.8.1.3.4 Procedure

In the following description, the references to Bridge Hello Time apply only to Bridges that support STP or RSTP.

The input parameter values are checked for compliance with their definitions in Clause 13. If they do not comply, or the value of Bridge Max Age or Bridge Forward Delay is less than the lower limit of the range specified in Table 13-5, then no action shall be taken for any of the supplied parameters. If the value of any of Bridge Max Age, Bridge Forward Delay, or Bridge Hello Time is outside the range specified in Table 13-5, then the Bridge need not take action.

Otherwise:

- a) The Bridge's Bridge Max Age, Bridge Hello Time, and Bridge Forward Delay parameters are set to the supplied values.
- b) In STP Bridges, the Set Bridge Priority procedure (8.8.4 of IEEE Std 802.1D, 1998 Edition) is used to set the priority part of the Bridge Identifier to the supplied value.
- c) In RSTP and MSTP Bridges, the priority component of the Bridge Identifier (13.24.1) is updated using the supplied value. For all Ports of the Bridge, the reselect for the CIST parameter (13.25) is set TRUE, and the selected parameter for the CIST (13.25) is set FALSE.

12.8.1.4 Set MSTI Bridge Protocol Parameters

12.8.1.4.1 Purpose

To modify parameters in the Bridge's Bridge Protocol Entity for the specified Spanning Tree instance, in order to force a configuration of the spanning tree and/or tune the reconfiguration time to suit a specific topology.

12.8.1.4.2 Inputs

- a) MSTID—identifies the set of parameters upon which the operation will be performed.
- b) Bridge Priority—the new value of the priority part of the Bridge Identifier (13.24.1) for the Spanning Tree instance identified by the MSTID.

12.8.1.4.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to invalid Bridge Priority value (12.3); or
 - 2) Operation rejected due to invalid MSTID (i.e., there is currently no MST Instance with that value of MSTID supported by the Bridge); or
 - 3) Operation accepted.

12.8.1.4.4 Procedure

The Bridge Priority parameter value is checked for compliance with its definition in Clause 13. If it does not comply, then no action shall be taken.

Otherwise, the priority part of the Bridge Identifier is set to the supplied value for the specified Spanning Tree instance.

12.8.2 Bridge Port

A Bridge Port object models the operations related to an individual Bridge Port in relation to the operation of the Spanning Tree Algorithm and Protocol. There are a fixed set of Bridge Ports per Bridge; each can, therefore, be identified by a permanently allocated Port Number, as a fixed component of the Protocol Entity resource.

The management operations that can be performed on a Bridge Port are

- a) Read CIST Port Parameters (12.8.2.1);
- b) Read MSTI Port Parameters (12.8.2.2);
- c) Set CIST Port Parameters (12.8.2.3);
- d) Set MSTI Port Parameters (12.8.2.4);
- e) Force BPDU Migration Check (12.8.2.5).

12.8.2.1 Read CIST Port Parameters

12.8.2.1.1 Purpose

To obtain information regarding a specific Port within the Bridge's Bridge Protocol Entity, for the CIST.

12.8.2.1.2 Inputs

- a) Port Number—the number of the Bridge Port.

12.8.2.1.3 Outputs

- a) Uptime—count in seconds of the time elapsed since the Port was last reset or initialized (BEGIN, 13.24).
- b) Port State—Discarding, Listening, Learning, or Forwarding (8.4).
- c) Port Identifier—the unique Port identifier comprising two parts, the Port Number and the Port Priority field (13.25.32).
- d) Path Cost (ExternalPortPathCost, 13.25.12).
- e) Designated Root (CIST Root Identifier, 13.25.7).
- f) Designated Cost (External Root Path Cost, 13.25.7).
- g) Designated Bridge (13.25.7).
- h) Designated Port (13.25.7).
- i) Topology Change Acknowledge (13.25.57).
- j) Hello Time (13.25.34).
- k) adminEdgePort (13.25.1). Present in implementations that support the identification of edge ports.
- l) operEdgePort (13.25.30). Present in implementations that support the identification of edge ports.
- m) autoEdgePort (13.25.5). Optional and provided only in implementations that support the automatic identification of edge ports.
- n) autoIsolatePort (13.25.6). Present in implementations that support detection of fragile bridges.
- o) isolatePort (13.25.20). Present in implementations that support detection of fragile bridges.
- p) MAC Enabled—the current state of the MAC Enabled parameter (6.6.2). Present if the implementation supports the MAC Enabled parameter.
- q) MAC Operational—the current state of the MAC Operational parameter (6.6.2). Present if the implementation supports the MAC Operational parameter.
- r) adminPointToPointMAC—the current state of the adminPointToPointMAC parameter (6.6.3). Present if the implementation supports the adminPointToPointMAC parameter.

- s) operPointToPointMAC—the current state of the operPointToPointMAC parameter (6.6.3). Present if the implementation supports the operPointToPointMAC parameter.
- t) restrictedRole—the current state of the restrictedRole parameter for the Port (13.25.49).
- u) restrictedTcn—the current state of the restrictedTcn parameter for the Port (13.25.50).
- v) Port Role—the current Port Role for the Port (i.e., Root, Alternate, Designated, or Backup)
- w) Disputed—the current value of the disputed variable for the CIST for the Port (13.25.9).
- x) enableBPDUrx—the value of enableBPDUrx (13.25.10).
- y) enableBPDUtx—the value of enableBPDUtx (13.25.11).
- z) pseudoRootId—the value of Bridge Identifier used by the L2 gateway protocol (13.25.37).
- aa) isL2gp—the value of isL2gp (13.25.19).

The following additional parameters are present only if the Bridge supports MSTP:

- ab) CIST Regional Root Identifier (13.24.8). The Bridge Identifier of the current CIST Regional Root.
- ac) CIST Path Cost (CIST Internal Root Path Cost, in 13.24.8). The CIST path cost from the transmitting Bridge to the CIST Regional Root.
- ad) Port Hello Time. The administrative value of Hello Time for the Port (13.25.34).

12.8.2.2 Read MSTI Port Parameters

12.8.2.2.1 Purpose

To obtain information regarding a specific Port within the Bridge’s Bridge Protocol Entity, for a given MSTI.

12.8.2.2.2 Inputs

- a) Port Number—the number of the Bridge Port.
- b) MSTID—identifies the set of parameters that will be returned, in the range 1 through 4094.

12.8.2.2.3 Outputs

- a) MSTID—identifies the set of parameters that are being returned. This parameter is the identifier of the spanning tree for which the operation is being performed.
- b) Uptime—count in seconds of the time elapsed since the Port was last reset or initialized (BEGIN).
- c) State—the current state of the Port (i.e., Disabled, Listening, Learning, Forwarding, or Blocking) (8.4, 13.36).
- d) Port Identifier—the unique Port identifier comprising two parts, the Port Number and the Port Priority field (13.25.32).
- e) Path Cost (13.25.7).
- f) Designated Root (13.25.7).
- g) Designated Cost (13.25.7).
- h) Designated Bridge (13.25.7).
- i) Designated Port (13.25.7).
- j) Port Role—the current Port Role for the Port (i.e., Root, Alternate, Designated, or Backup, 13.25.51)
- k) Disputed—the current value of the disputed variable (13.25.9).
- l) pseudoRootId—the new value of Bridge Identifier used by the L2 gateway protocol (13.25.37).

12.8.2.3 Set CIST port parameters

12.8.2.3.1 Purpose

To modify parameters for a Port in the Bridge's Bridge Protocol Entity in order to force a configuration of the spanning tree for the CIST.

12.8.2.3.2 Inputs

- a) Port Number—the number of the Bridge Port.
- b) Path Cost—the new ExternalPortPathCost (13.25.12).
- c) Port Priority—the new value of the priority field for the Port Identifier (13.25.32).
- d) adminEdgePort—the new value of the adminEdgePort parameter (13.25.1). Present in implementations that support the identification of edge ports.
- e) autoEdgePort—the new value of the autoEdgePort parameter (13.25.5). Optional and provided only in implementations that support the automatic identification of edge ports. .
- f) autoIsolatePort—the new value of the autoIsolatePort parameter (13.25.6). Present in implementations that support detection of fragile bridges.
- g) MAC Enabled—the new value of the MAC Enabled parameter (6.6.2). May be present if the implementation supports the MAC Enabled parameter.
- h) adminPointToPointMAC—the new value of the adminPointToPointMAC parameter (6.6.3). May be present if the implementation supports the adminPointToPointMAC parameter.
- i) restrictedRole—the new value of the restrictedRole parameter for the Port (13.25.49).
- j) restrictedTcn—the new value of the restrictedTcn parameter for the Port (13.25.50).
- k) enableBPDURx—the new value of enableBPDURx (13.25.10).
- l) enableBPDUtx—the new value of enableBPDUtx (13.25.11).
- m) pseudoRootId—the new value of Bridge Identifier used by the L2 gateway protocol (13.25.37).
- n) isL2gp—the new value of isL2gp (13.25.19).

12.8.2.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to invalid Port Priority value (12.3); or
 - 2) Operation accepted.

12.8.2.3.4 Procedure

In STP Bridges, the Set Path Cost procedure (8.8.6 of IEEE Std 802.1D, 1998 Edition) is used to set the Path Cost parameter for the specified Port for the specified spanning tree instance. The Set Port Priority procedure (8.8.5 of IEEE Std 802.1D, 1998 Edition) is used to set the priority part of the Port Identifier (8.5.5.1 of IEEE Std 802.1D, 1998 Edition) for the CIST to the supplied value.

In RSTP and MSTP Bridges, the Path Cost (13.25.12) and Port Priority (13.25.33) parameters for the Port are updated using the supplied values. The reselect parameter value for the CIST for the Port (13.25.48) is set TRUE, and the selected parameter for the CIST for the Port (13.25.52) is set FALSE.

12.8.2.4 Set MSTI port parameters

12.8.2.4.1 Purpose

To modify parameters for a Port in the Bridge's Bridge Protocol Entity in order to force a configuration of the spanning tree for the specified Spanning Tree instance.

12.8.2.4.2 Inputs

- a) MSTID—identifies the set of parameters upon which the operation will be performed. This parameter is the identifier of the spanning tree for which the operation is being performed.
- b) Port Number—the number of the Bridge Port.
- c) Path Cost—the new value (13.25.12).
- d) Port Priority—the new value of the priority field for the Port Identifier (13.25.32).
- e) pseudoRootId—the new value of Bridge Identifier used by the L2 gateway protocol (13.25.37).

12.8.2.4.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to invalid Port Priority value (12.3); or
 - 2) Operation rejected due to invalid MSTID (i.e., there is currently no spanning tree instance with that value of MSTID supported by the Bridge); or
 - 3) Operation accepted.

12.8.2.4.4 Procedure

The Path Cost (13.25.12) and Port Priority (13.25.33) parameters for the specified MSTI and Port are updated using the supplied values. The reselect parameter value for the MSTI for the Port (13.25.48) is set TRUE, and the selected parameter for the MSTI for the Port (13.25.52) is set FALSE.

12.8.2.5 Force BPDU Migration Check

This operation is available only in Bridges that support RSTP or MSTP, as specified in Clause 13.

12.8.2.5.1 Purpose

To force the specified Port to transmit RST or MST BPDUs (see 13.30).

12.8.2.5.2 Inputs

- a) Port Number—the number of the Bridge Port.

12.8.2.5.3 Outputs

None.

12.8.2.5.4 Procedure

The mcheck variable (13.25.25) for the specified Port is set to the value TRUE if the value of the forceVersion variable (13.6.2) is greater than or equal to 2.

12.9 MRP Entities

The operation of MRP is described in Clause 10.

The objects that comprise this managed resource are

- a) The MRP Timer objects (12.9.1);
- b) The MRP Attribute Type objects (12.9.2);
- c) The Periodic State Machine objects (12.9.3).

12.9.1 The MRP Timer object

The MRP Timer object models the operations that can be performed on, or inquire about, the current settings of the timers used by MRP on a given Port. The management operations that can be performed on the MRP Participant are

- a) Read MRP Timers (12.9.1.1);
- b) Set MRP Timers (12.9.1.2).

NOTE—The MRP timer values modeled by this object are the values used to initialize timer instances that are used within the MRP state machines, not the timer instances themselves. Hence, there is a single MRP Timer object per Port, regardless of whether the Bridge supports single or multiple spanning trees.

12.9.1.1 Read MRP Timers

12.9.1.1.1 Purpose

To read the current MRP Timers for a given Port.

12.9.1.1.2 Inputs

- a) The Port identifier.

12.9.1.1.3 Outputs

- a) Current value of JoinTime—Centiseconds, *Persistent*. (10.7.4.1 and 10.7.11);
- b) Current value of LeaveTime—Centiseconds, *Persistent*. (10.7.4.2 and 10.7.11);
- c) Current value of LeaveAllTime—Centiseconds, *Persistent*. (10.7.4.3 and 10.7.11).

12.9.1.2 Set MRP Timers

12.9.1.2.1 Purpose

To set new values for the MRP Timers for a given Port.

12.9.1.2.2 Inputs

- a) The Port identifier;
- b) New value of JoinTime—Centiseconds (10.7.4.1 and 10.7.11);
- c) New value of LeaveTime—Centiseconds (10.7.4.2 and 10.7.11);
- d) New value of LeaveAllTime—Centiseconds (10.7.4.3 and 10.7.11).

12.9.1.2.3 Outputs

None.

12.9.2 The MRP Attribute Type object

The MRP Attribute Type object models the operations that can be performed on, or inquire about, the operation of MRP for a given Attribute Type (10.8.2.2). The management operations that can be performed on an MRP Attribute Type are

- a) Read MRP Applicant Controls (12.9.2.1);
- b) Set MRP Applicant Controls (12.9.2.2).

12.9.2.1 Read MRP Applicant Controls

12.9.2.1.1 Purpose

To read the current values of the MRP Applicant Administrative control parameters (10.8.2.2) associated with all MRP Participants for a given Port, MRP Application, and Attribute Type.

12.9.2.1.2 Inputs

- a) The Port identifier;
- b) The MRP Application address (Table 10-1);
- c) The Attribute Type (10.8.2.2).

12.9.2.1.3 Outputs

- a) The current Applicant Administrative Control Value, *Persistent*. (10.7.3);
- b) Failed Registrations—count of the number of times that this MRP Application has failed to register an attribute of this type due to lack of space in the Filtering Database, *Persistent*. (12.10.1.6).

12.9.2.2 Set MRP Applicant Controls

12.9.2.2.1 Purpose

To set new values for the MRP Applicant Administrative control parameters (10.7.3) associated with all MRP Participants for a given Port, MRP Application, and Attribute Type.

12.9.2.2.2 Inputs

- a) The Port identifier;
- b) The MRP Application address (Table 10-1);
- c) The Attribute Type (10.8.2.2) associated with the state machine;
- d) The desired Applicant Administrative Control Value (10.7.3).

12.9.2.2.3 Outputs

None.

12.9.3 Periodic state machine objects

The Periodic state machine object models the operations that can be performed upon, or inquire about, the operation of the Periodic state machine for a given Port.

The management operations that can be performed on a Periodic state machine are Read Periodic state machine state and Set Periodic state machine state.

12.9.3.1 Read Periodic state machine state

12.9.3.1.1 Purpose

To inquire whether a particular Periodic state machine is enabled or disabled.

12.9.3.1.2 Inputs

- a) The Port identifier.

12.9.3.1.3 Outputs

- a) The Port identifier.
- b) The state of the Periodic state machine. This can take either of the values “enabled” or “disabled.”

12.9.3.2 Set Periodic state machine state

12.9.3.2.1 Purpose

To enable or disable a particular Periodic state machine.

12.9.3.2.2 Inputs

- a) The Port identifier.
- b) The desired state of the Periodic state machine. This can take either of the values “enabled” or “disabled.”

12.9.3.2.3 Outputs

None.

12.10 Bridge VLAN managed objects

The following managed objects define the semantics of the management operations that can be performed on the VLAN aspects of a Bridge:

- a) The Bridge VLAN Configuration managed object (12.10.1);
- b) The VLAN Configuration managed object (12.10.2);
- c) The VLAN Learning Constraints managed object (12.10.3).

12.10.1 Bridge VLAN Configuration managed object

The Bridge VLAN Configuration managed object models operations that modify, or inquire about, the overall configuration of the Bridge’s VLAN resources. There is a single Bridge VLAN Configuration managed object per Bridge.

The management operations that can be performed on the Bridge VLAN Configuration managed object are

- a) Read Bridge VLAN Configuration (12.10.1.1);
- b) Configure PVID and VID Set values (12.10.1.2);
- c) Configure Acceptable Frame Types parameters (12.10.1.3);
- d) Configure Enable Ingress Filtering parameters (12.10.1.4);
- e) Reset Bridge (12.10.1.5);
- f) Configure Restricted_VLAN_Registration parameters (12.10.1.6);
- g) Configure Protocol Group Database (12.10.1.7);
- h) Configure VLAN Learning Constraints (12.10.3).

12.10.1.1 Read Bridge VLAN Configuration

12.10.1.1.1 Purpose

To obtain general VLAN information from a Bridge.

12.10.1.1.2 Inputs

None.

12.10.1.1.3 Outputs

- a) The IEEE 802.1Q VLAN Version number. Reported as “1” by Bridges that support only SST operation, and reported as “2” by Bridges that support MST operation;

NOTE—No IEEE 802.1Q VLAN version numbers other than 1 and 2 are currently specified.

- b) The optional VLAN features supported by the implementation:
 - 1) The maximum number of VIDs supported;
 - 2) Whether the implementation supports the ability to override the default PVID setting, and its egress status (VLAN-tagged or untagged) on each Port;
 - 3) For a Bridge that supports Port-and-Protocol-based VLAN classification, which of the Protocol Template formats (6.12.1) are supported by the implementation.
 - 4) For MST Bridges, the maximum number of MSTIs supported within an MST Region (i.e., the number of Spanning Tree instances that can be supported in addition to the CIST). For SST Bridges, this parameter may be either omitted or reported as “0.”
- c) For each Port:
 - 1) The Port number;
 - 2) The PVID value (6.9) currently assigned to that Port;
 - 3) For a Bridge that supports Port-and-Protocol-based VLAN classification, whether the implementation supports Port-and-Protocol-based VLAN classification on that Port;
 - 4) For a Bridge that supports Port-and-Protocol-based VLAN classification on that Port, the maximum number of entries supported in the VID Set on that Port; the VID value and Protocol Group Identifier currently assigned to each entry in the VID Set (8.6.2) on that Port;
 - 5) The state of the Acceptable Frame Types parameter (6.9). The permissible values for this parameter are:
 - i) *Admit Only VLAN-tagged frames*;
 - ii) *Admit Only Untagged and Priority Tagged frames*;
 - iii) *Admit All frames*.
 - 4) The state of the Enable Ingress Filtering parameter (6.9); Enabled or Disabled;
 - 5) The state of the Restricted_VLAN_Registration parameter (11.2.3.2.3), TRUE or FALSE.
- d) For a Bridge that supports Port-and-Protocol-based VLAN classification: the contents of the Protocol Group Database comprising a set of {Protocol Template, Protocol Group Identifier} bindings (6.12.1, 6.12.2, and 6.12.3); the maximum number of entries supported in the Protocol Group Database.

12.10.1.2 Configure PVID and VID Set values

12.10.1.2.1 Purpose

To configure the PVID and VID Set value(s) (6.9) associated with one or more Ports.

12.10.1.2.2 Inputs

- a) For each Port to be configured, a Port number and the PVID value to be associated with that Port
- b) In addition, for a Bridge that supports Port-and-Protocol-based VLAN classification: for each Port to be configured, a Port number, a Protocol Group Identifier, and a VID value for the member of the Port’s VID Set that is to be configured.

12.10.1.2.3 Outputs

- a) Operation status for each Port to be configured. This takes one of the following values:
 - 1) Operation rejected due to there being no spare VID Set entries on this Port; or
 - 2) Operation rejected due to the PVID or VID being out of the supported range for this Port; or
 - 3) Operation accepted.

12.10.1.3 Configure Acceptable Frame Types parameters

12.10.1.3.1 Purpose

To configure the Acceptable Frame Types parameter (6.9) associated with one or more Ports.

12.10.1.3.2 Inputs

- a) For each Port to be configured, a Port number and the value of the Acceptable Frame Types parameter to be associated with that Port. The permissible values of this parameter are (as defined in 6.9):
 - 1) *Admit Only VLAN Tagged frames*;
 - 2) *Admit Only Untagged and Priority Tagged frames*
 - 3) *Admit All frames*.

12.10.1.3.3 Outputs

None.

12.10.1.4 Configure Enable Ingress Filtering parameters

12.10.1.4.1 Purpose

To configure the Enable Ingress Filtering parameter(s) (8.6.2) associated with one or more Ports.

12.10.1.4.2 Inputs

- a) For each Port to be configured, a Port number and the value of the Enable Ingress Filtering parameter to be associated with that Port. The permissible values for the parameter are
 - 1) Enabled;
 - 2) Disabled.

12.10.1.4.3 Outputs

None.

12.10.1.5 Reset Bridge

12.10.1.5.1 Purpose

To reset all statically configured VLAN-related information in the Bridge to its default state. This operation

- a) Deletes all VLAN Configuration managed objects;
- b) Resets the PVID associated with each Bridge Port to the Default PVID value (Table 9-2);
- c) Removes all entries in the Protocol Group Database and removes all members of the VID Set on each port, for a Bridge that supports Port-and-Protocol-based VLAN classification;

- d) Resets the Acceptable Frame Types parameter value associated with each Port to the default value (6.9).

12.10.1.5.2 Inputs

None.

12.10.1.5.3 Outputs

None.

12.10.1.6 Configure Restricted_VLAN_Registration parameters

12.10.1.6.1 Purpose

To configure the Restricted_VLAN_Registration parameter (11.2.3.2.3) associated with one or more Ports.

12.10.1.6.2 Inputs

- a) For each Port to be configured, a Port number and the value of the Restricted_VLAN_Registration parameter. The permissible values of this parameter are (as defined in 11.2.3.2.3) as follows:
 - 1) TRUE;
 - 2) FALSE.

12.10.1.6.3 Outputs

None.

12.10.1.7 Configure Protocol Group Database

To configure a Protocol Group Database (6.12.3) entry. This operation is not applicable to a Bridge that does not support Port-and-Protocol-based VLAN classification.

NOTE—Implementation of the Configure Protocol Group Database operation is not mandatory; conformant implementations may implement a fixed set of Protocol Group Database entries.

12.10.1.7.1 Inputs

- a) A value representing the frame format to be matched: Ethernet, RFC_1042, SNAP_8021H, SNAP_Other or LLC_Other (6.12.1);
- b) One of
 - 1) An IEEE 802.3 EtherType value, for matching frame formats of Ethernet, RFC_1042, or SNAP_8021H;
 - 2) A 40-bit Protocol ID (PID), for matching frame formats of SNAP_Other;
 - 3) A pair of ISO/IEC 8802-2 DSAP and SSAP address field values, for matching frame formats of LLC_Other;
- c) A Protocol Group Identifier (6.12.2).

NOTE—While the intent is to identify VID traffic, an ID of 0 may be used to indicate untagged traffic that needs prioritization.

12.10.1.7.2 Outputs

- a) Operation status. This takes one of the following values:

- 1) Operation rejected due to there being no spare Protocol Group Database entries; or
- 2) Operation rejected due to an unsupported frame format; or
- 3) Operation rejected due to an unsupported value for an IEEE 802.3 EtherType value, PID, DSAP, or SSAP; or
- 4) Operation accepted.

12.10.2 VLAN Configuration managed object

The VLAN Configuration object models operations that modify, or inquire about, the configuration of a particular VID within a Bridge. There are multiple VLAN Configuration objects per Bridge; only one such object can exist for a given VLAN ID.

The management operations that can be performed on the VLAN Configuration are:

- a) Read VLAN Configuration (12.10.2.1);
- b) Create VLAN Configuration (12.10.2.2);
- c) Delete VLAN Configuration (12.10.2.3).

12.10.2.1 Read VLAN Configuration

12.10.2.1.1 Purpose

To obtain general information regarding a specific VLAN Configuration.

12.10.2.1.2 Inputs

- a) VLAN Identifier: a 12-bit VID.

12.10.2.1.3 Outputs

- a) VLAN Name: A text string of up to 32 characters of locally determined significance;
- b) List of Untagged Ports: The set of Port numbers in the untagged set (8.8.2) for this VLAN ID;
- c) List of Egress Ports: The set of Port numbers in the member set (8.8.10) for this VLAN ID.

NOTE—The values of the member set and the untagged set are determined by the values held in VLAN Registration Entries in the Filtering Database (8.8.2, 8.8.5, and 8.8.10).

12.10.2.2 Create VLAN Configuration

12.10.2.2.1 Purpose

To create or update a VLAN Configuration managed object.

12.10.2.2.2 Inputs

- a) VLAN Identifier: a 12-bit VID;
- b) VLAN Name: a text string of up to 32 characters of locally determined significance.

NOTE—Static configuration of the member set and the Untagged set is achieved by means of the management operations for manipulation of VLAN Registration Entries (12.7.5).

12.10.2.2.3 Outputs

None.

12.10.2.3 Delete VLAN Configuration

12.10.2.3.1 Purpose

To delete a VLAN Configuration managed object.

12.10.2.3.2 Inputs

- a) VLAN Identifier: a 12-bit VID;

12.10.2.3.3 Outputs

None.

12.10.3 The VLAN Learning Constraints managed object

The VLAN Learning Constraints managed object models operations that modify, or inquire about, the set of VLAN Learning Constraints (8.8.8.2) and VID to FID allocations (8.8.8.1) that apply to the operation of the Learning Process and the Filtering Database. There is a single VLAN Learning Constraints managed object per Bridge. The object is modeled as a pair of fixed-length tables, as follows:

- a) A Learning Constraint table in which each table entry either defines a single Learning Constraint or is undefined. For some of the operations that can be performed on the table, an *entry index* is used; this identifies the number of the entry in the table, where index number 1 is the first, and N is the last (where the table contains N entries).

NOTE—The number of Learning Constraint table entries supported is an implementation option. This standard does not provide any distribution mechanism to ensure that the same set of constraints is configured in all Bridges; individual Bridges can be configured by use of the management operations defined in this subclause (for example, via the use of SNMP operating on a Bridge MIB), but there is no in-built consistency checking to ensure that all Bridges have been provided with the same constraint information. Hence, any such consistency checking is the responsibility of the network administrator and the management applications employed in the LAN.

- b) A VID to FID allocation table (8.8.8.1) with an entry per VID supported by the implementation. Each table entry indicates, for that VID, that there is currently
 - 1) No allocation defined; or
 - 2) A fixed allocation to FID X; or
 - 3) A dynamic allocation to FID X.

The management operations that can be performed on the VLAN Learning Constraints managed object are

- c) Read VLAN Learning Constraints (12.10.3.1);
- d) Read VLAN Learning Constraints for VID (12.10.3.2);
- e) Set VLAN Learning Constraint (12.10.3.3);
- f) Delete VLAN Learning Constraint (12.10.3.4);
- g) Read VID to FID allocations (12.10.3.5);
- h) Read FID allocation for VID (12.10.3.6);
- i) Read VIDs allocated to FID (12.10.3.7);
- j) Set VID to FID allocation (12.10.3.8);
- k) Delete VID to FID allocation (12.10.3.9).

12.10.3.1 Read VLAN Learning Constraints

12.10.3.1.1 Purpose

To read the contents of a range of one or more entries in the VLAN Learning Constraints table.

12.10.3.1.2 Inputs

- a) First Entry—Entry Index of first entry to be read;
- b) Last Entry—Entry Index of last entry to be read.

12.10.3.1.3 Outputs

- a) List of Entries—for each entry that was read:
 - 1) The Entry Index;
 - 2) The type of the Learning Constraint: Undefined, S or I;
 - 3) The value of the Learning Constraint, which is one of:
 - i) Undefined, indicating an empty element in the table;
 - ii) An S Constraint value, consisting of a pair of VIDs;
 - iii) An I Constraint value, consisting of a VID and an Independent Set Identifier.

NOTE—Where this operation is implemented using a remote management protocol, PDU size constraints may restrict the number of entries that are actually read to fewer than was requested in the input parameters. In such cases, retrieving the remainder of the desired entry range can be achieved by repeating the operation with a modified entry range specification.

12.10.3.2 Read VLAN Learning Constraints for VID

12.10.3.2.1 Purpose

To read all the VLAN Learning Constraints for a given VID.

12.10.3.2.2 Inputs

- a) VID—The VLAN Identifier to which the read operation applies.

12.10.3.2.3 Outputs

- a) All learning constraint values that identify the VID requested. Each value returned is either
 - 1) An S Constraint value, consisting of a pair of VIDs; or
 - 2) An I Constraint value, consisting of a VID and an Independent Set Identifier.

12.10.3.3 Set VLAN Learning Constraint

12.10.3.3.1 Purpose

To modify the contents of one of the entries in the VLAN Learning Constraints table.

12.10.3.3.2 Inputs

- a) Entry Index—Entry index of the entry to be set;
- b) The type of the Learning Constraint: S or I;
- c) The value of the Learning Constraint, which is either:
 - 1) An S Constraint value, consisting of a pair of VIDs; or

- 2) An I Constraint value, consisting of a VID and an Independent Set Identifier.

12.10.3.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to inconsistent learning constraint specification (8.8.8.3)—The Set operation requested setting a constraint that is inconsistent with another constraint already defined in the constraint table. The operation returns the value of the constraint concerned; or
 - 2) Operation rejected due to inconsistent fixed VID to FID allocation (8.8.8.3)—The Set operation requested setting a constraint that is inconsistent with a fixed VID to FID allocation already defined in the allocation table. The operation returns the value of the fixed allocation concerned; or
 - 3) Operation rejected due to entry index exceeding the maximum index supported by the constraint table; or
 - 4) Operation rejected due to conflict with FID to MSTID allocations (12.12.2)—The Set operation requested setting a constraint that cannot be reconciled with the current FID to MSTID allocations represented by the FID to MSTID Allocation Table; or

NOTE—It is not possible to specify a shared VLAN learning constraint for VIDs that do not share the same Spanning Tree instance.

- 5) Operation accepted.

12.10.3.3.4 Procedure

In MST Bridges, the Configuration Digest element of the MST Configuration Identifier is recalculated, in accordance with the definition in 13.7, following any change in the VLAN Learning Constraints that results in a change in the allocation of VIDs to spanning trees.

12.10.3.4 Delete VLAN Learning Constraint

12.10.3.4.1 Purpose

To remove one of the entries in the VLAN Learning Constraints table. This operation has the effect of setting the value of the specified table entry to “Undefined.”

12.10.3.4.2 Inputs

- a) Entry Index—Entry index of the entry to be deleted.

12.10.3.4.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to entry index exceeding the maximum index supported by the constraint table; or
 - 2) Operation accepted.

12.10.3.4.4 Procedure

In MST Bridges, the Configuration Digest element of the MST Configuration Identifier is recalculated, in accordance with the definition in 13.7, following any change in the VLAN Learning Constraints that results in a change in the allocation of VIDs to spanning trees.

12.10.3.5 Read VID to FID allocations

12.10.3.5.1 Purpose

To read the contents of a range of one or more entries in the VID to FID allocation table.

12.10.3.5.2 Inputs

- a) First Entry—VID of first entry to be read;
- b) Last Entry—VID of last entry to be read.

12.10.3.5.3 Outputs

- a) List of Entries—For each entry that was read:
 - 1) VID—the VLAN Identifier for this entry;
 - 2) Allocation Type—the type of the allocation: Undefined, Fixed or Dynamic;
 - 3) FID—the FID to which the VID is allocated (if not of type Undefined).

NOTE—Where this operation is implemented using a remote management protocol, PDU size constraints may restrict the number of entries that are actually read to fewer than was requested in the input parameters. In such cases, retrieving the remainder of the desired entry range can be achieved by repeating the operation with a modified entry range specification.

12.10.3.6 Read FID allocation for VID

12.10.3.6.1 Purpose

To read the FID to which a specified VID is currently allocated.

12.10.3.6.2 Inputs

- a) VID—the VLAN Identifier to which the read operation applies.

12.10.3.6.3 Outputs

- a) VID—the VLAN Identifier to which the read operation applies;
- b) Allocation Type—the type of the allocation: Undefined, Fixed or Dynamic;
- c) FID—the FID to which the VID is allocated (if not of type Undefined).

12.10.3.7 Read VIDs allocated to FID

12.10.3.7.1 Purpose

To read all VIDs currently allocated to a given FID.

12.10.3.7.2 Inputs

- a) FID—the Filtering Identifier to which the read operation applies.

12.10.3.7.3 Outputs

- a) FID—the Filtering Identifier to which the read operation applies
- b) Allocation List—a list of allocations for this FID. For each element in the list:
 - 1) Allocation Type—the type of the allocation: Fixed or Dynamic;

- 2) VID—the VID that is allocated.

12.10.3.8 Set VID to FID allocation

12.10.3.8.1 Purpose

To establish a fixed allocation of a VID to an FID.

12.10.3.8.2 Inputs

- a) VID—the VID of the entry to be set;
- b) FID—the FID to which the VID is to be allocated.

12.10.3.8.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to inconsistent learning constraint specification (8.8.8.3)—the Set operation requested setting a fixed allocation that is inconsistent with a VLAN Learning Constraint. The operation returns the value of the VLAN Learning Constraint concerned; or
 - 2) Operation rejected due to VID exceeding the maximum VID supported by the allocation table; or
 - 3) Operation rejected due to FID exceeding the maximum ID supported by the implementation; or
 - 4) Operation accepted.

12.10.3.8.4 Procedure

In MST Bridges, the Configuration Digest element of the MST Configuration Identifier is recalculated, in accordance with the definition in 13.7, following any change in the allocation of VIDs to FIDs that results in a change in the allocation of VIDs to spanning trees.

12.10.3.9 Delete VID to FID allocation

12.10.3.9.1 Purpose

To remove a fixed VID to FID allocation from the VID to FID allocation table. This operation has the effect of setting the value of the specified table entry to “Undefined.”

NOTE—If the VID concerned represents a currently active VID, then removal of a fixed allocation may result in the “Undefined” value in the table immediately being replaced by a dynamic allocation to an FID.

12.10.3.9.2 Inputs

- a) VID—VID of the allocation to be deleted.

12.10.3.9.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to VID exceeding the maximum value supported by the allocation table; or
 - 2) Operation accepted.

12.10.3.9.4 Procedure

In MST Bridges, the Configuration Digest element of the MST Configuration Identifier is recalculated, in accordance with the definition in 13.7, and the MSTP state machine variables are reinitialized by asserting BEGIN, following any change in the allocation of VIDs to FIDs that results in a change in the allocation of VIDs to spanning trees.

12.11 MMRP entities

The following managed objects define the semantics of the management operations that can be performed on the operation of MMRP in a Bridge:

- a) The MMRP Configuration managed object (12.10.1).

12.11.1 MMRP Configuration managed object

The MMRP Configuration managed object models operations that modify, or inquire about, the overall configuration of the operation of MMRP. There is a single MMRP Configuration managed object per Bridge.

The management operations that can be performed on the MMRP Configuration managed object are as follows:

- a) Read MMRP Configuration (12.10.1.1);
- b) Notify MAC Address registration failure (12.10.1.6);
- c) Configure Restricted_MAC_Address_Registration parameters (12.11.1.3).

12.11.1.1 Read MMRP Configuration

12.11.1.1.1 Purpose

To obtain general MMRP configuration information from a Bridge.

12.11.1.1.2 Inputs

None.

12.11.1.1.3 Outputs

- a) For each Port:
 - 1) The Port number;
 - 2) The state of the Restricted_MAC_Address_Registration parameter, *Persistent*. (10.12.2.3), TRUE or FALSE.

12.11.1.2 Notify MAC Address registration failure

12.11.1.2.1 Purpose

To notify a manager that MMRP has failed to register a given MAC Address owing to lack of resources in the Filtering Database for the creation of a MAC Address Registration Entry (8.8.4) or to the Restricted_MAC_Address_Registration parameter.

12.11.1.2.2 Inputs

None.

12.11.1.2.3 Outputs

- a) The MAC address that MMRP failed to register;
- b) The Port number of the Port on which the registration request was received.
- c) The reason for the failure:
 - 1) Lack of Resources; or
 - 2) Registration Restricted.

12.11.1.3 Configure Restricted_MAC_Address_Registration parameters

12.11.1.3.1 Purpose

To configure the Restricted_MAC_Address_Registration parameter (10.12.2.3) associated with one or more Ports.

12.11.1.3.2 Inputs

- a) For each Port to be configured, a Port number and the value of the Restricted_MAC_Address_Registration parameter. The permissible values of this parameter are (as defined in 10.12.2.3):
 - 1) TRUE;
 - 2) FALSE.

12.11.1.3.3 Outputs

None.

12.12 MST configuration entities

The following managed objects define the semantics of the management operations that can be performed on the MST configuration in a Bridge:

- a) The MSTI List object (12.12.1);
- b) The FID to MSTID Allocation Table object (12.12.2);
- c) The MST Configuration Table object (12.12.3).

12.12.1 The MSTI List

For MST Bridges, the MSTI List object models the operations that modify, or inquire about, the list of MST spanning tree instances supported by the Bridge. The object is modeled as a list of MSTIDs corresponding to the MSTIs supported by the Bridge.

The MSTID List object supports the following operations:

- a) Read MSTI List (12.12.1.1);
- b) Create MSTI (12.12.1.2);
- c) Delete MSTI (12.12.1.3).

12.12.1.1 Read MSTI List

12.12.1.1.1 Purpose

To read the list of MSTIDs that are currently supported by the Bridge.

12.12.1.1.2 Inputs

None.

12.12.1.1.3 Outputs

- a) MSTID list. The list of MSTID values that are currently supported by the Bridge.

12.12.1.2 Create MSTI

12.12.1.2.1 Purpose

To create a new MSTI and its associated state machines and parameters, and to add its MSTID to the MSTI List.

12.12.1.2.2 Inputs

- a) The MSTID of the MSTI to be created.

12.12.1.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to the number of MSTIs currently supported by the Bridge being equal to the maximum number of MSTIs that the Bridge is able to support.
 - 2) Operation rejected as the MSTID value supplied in the input parameters is already present in the MSTI List.
 - 3) Operation accepted.

12.12.1.3 Delete MSTI

12.12.1.3.1 Purpose

To delete an existing MSTI and its associated state machines and parameters, and to remove its MSTID from the MSTI List.

12.12.1.3.2 Inputs

- a) The MSTID of the MSTI to be deleted.

12.12.1.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected as the MSTID value supplied in the input parameters is not present in the MSTI List.
 - 2) Operation rejected as the MSTID value supplied in the input parameter currently has one or more FIDs allocated to it in the FID to MSTID Allocation Table.
 - 3) Operation accepted.

12.12.2 The FID to MSTID Allocation Table

For MST Bridges, the FID to MSTID Allocation Table object models the operations that modify, or inquire about, the assignment of FIDs to spanning tree instances currently supported by the Bridge (8.9.3). The object is modeled as a fixed-length table in which each entry in the table corresponds to a FID, and the value of the entry specifies the MSTID of the spanning tree to which the set of VIDs supported by that FID are assigned. A value of zero in an entry specifies that the set of VIDs supported by that FID are assigned to the CST.

The MSTID Allocation Table object supports the following operations:

- a) Read FID to MSTID allocations (12.12.2.1);
- b) Set FID to MSTID allocation (12.12.2.2).

12.12.2.1 Read FID to MSTID allocations

12.12.2.1.1 Purpose

To read a range of one or more entries in the FID to MSTID Allocation Table.

12.12.2.1.2 Inputs

- a) First FID—the FID of the first entry to be read;
- b) Last FID—the FID of the last entry to be read.

If the value of Last FID is numerically equal to, or smaller than, the value of First FID, then a single table entry is read, corresponding to the value of First FID.

12.12.2.1.3 Outputs

- a) List of entries—for each entry that was read:
 - 1) The FID of the entry; and
 - 2) The MSTID to which that FID is allocated.

12.12.2.2 Set FID to MSTID allocation

12.12.2.2.1 Purpose

To change the contents of an entry in the FID to MSTID Allocation Table.

12.12.2.2.2 Inputs

- a) FID—the FID of the entry to be changed;
- b) MSTID—the MSTID to which the FID is to be allocated.

12.12.2.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected as the MSTID value supplied in the input parameters is not present in the MSTI List.
 - 2) Operation rejected as the FID value supplied in the input parameters is invalid or is not supported.
 - 3) Operation accepted.

12.12.2.2.4 Procedure

The Configuration Digest element of the MST Configuration Identifier is recalculated, in accordance with the definition in 13.7, following any change in the allocations of FIDs to MSTIDs.

12.12.3 The MST Configuration Table

The MST Configuration Table managed object models the operations that can be performed on the MST Configuration Table for the Bridge (3.112, 8.9.1, and 13.7). Associated with the table is the MST Configuration Identifier for the Bridge (8.9.2, 13.7).

The MST Configuration Table is a read-only table, its elements derived from other configuration information held by the Bridge; specifically, the current state of the VID to FID allocation table (8.8.8.1, 12.10.3), and the FID to MSTID allocation table (8.9.3, 12.12.2). Hence, changes made to either of these tables can in turn affect the contents of the MST Configuration Table, and also affect the value of the “digest” element of the MST Configuration Identifier. The MST Configuration Table is modeled as a fixed table of 4096 elements, as described in 13.7.

The MST Configuration Table managed object supports the following operations:

- a) Read MST Configuration Table Element (12.12.3.1);
- b) Read VIDs assigned to MSTID (12.12.3.2);
- c) Read MST Configuration Identifier (12.12.3.3);
- d) Set MST Configuration Identifier Elements (12.12.3.4).

12.12.3.1 Read MST Configuration Table Element

12.12.3.1.1 Purpose

To read a single element of the current MST Configuration Table for the Bridge (13.7).

12.12.3.1.2 Inputs

- a) A VID value, in the range 0 through 4094.

12.12.3.1.3 Outputs

- a) A VID value, in the range 0 through 4094;
- b) The MSTID value corresponding to that VID.

12.12.3.2 Read VIDs assigned to MSTID

12.12.3.2.1 Purpose

To read the list of VIDs that are currently assigned to a given MSTID in the MST Configuration Table for the Bridge (13.7).

12.12.3.2.2 Inputs

- a) An MSTID value, in the range 0 through 4094.

12.12.3.2.3 Outputs

- a) A MSTID value, in the range 0 through 4094;

- b) A 4096-bit vector in which bit N is set TRUE if VID N is assigned to the given MSTID and is otherwise set FALSE.

12.12.3.3 Read MST Configuration Identifier

12.12.3.3.1 Purpose

To read the current value of the MST Configuration Identifier for the Bridge (13.7).

12.12.3.3.2 Inputs

None.

12.12.3.3.3 Outputs

- a) The MST Configuration Identifier (13.7), consisting of:
 - 1) The Configuration Identifier Format Selector in use by the Bridge. This has a value of 0 to indicate the format specified in this standard.
 - 2) The Configuration Name;
 - 3) The Revision Level;
 - 4) The Configuration Digest.

12.12.3.4 Set MST Configuration Identifier Elements

12.12.3.4.1 Purpose

To change the current values of the modifiable elements of the MST Configuration Identifier for the Bridge (13.7).

NOTE—The Configuration Digest element of the MST Configuration Identifier is read-only; its value is recalculated whenever configuration changes occur that result in a change in the allocation of VIDs to MSTIs.

12.12.3.4.2 Inputs

- a) The MST Configuration Identifier (13.7) Format Selector in use by the Bridge. This has a value of 0 to indicate the format specified in this standard.
- b) The Configuration Name;
- c) The Revision Level.

12.12.3.4.3 Outputs

- a) Operation Status. This can take the following values:
 - 1) Operation rejected due to unsupported Configuration Identifier Format Selector value.
 - 2) Operation accepted.

12.13 Provider Bridge management

The conformance requirements of Provider Bridges are specified in 5.10. The S-VLAN component and the externally accessible ports of all Provider Bridges, including Provider Edge Bridges, are managed using the managed objects defined in 12.4 through 12.12. This subclause defines additional managed objects specific to the operation of Provider Bridges.

The internal ports, LANs, and C-VLAN components of a Provider Edge Bridge are not managed directly using the managed objects defined in 12.4 through 12.12. Their operation is controlled and monitored through managed objects defined in this subclause.

Each externally accessible Bridge Port on a Provider Bridge is designated as a Provider Network Port, Customer Network Port, or Customer Edge Port. Designating a port as a Customer Edge Port implies Provider Edge Bridge functionality and, specifically, the existence of a C-VLAN component associated with that port. This C-VLAN component is uniquely identified within the Bridge by the port number of the associated Customer Edge Port. The management of the forwarding process, filtering data base, and C-VLANs of the C-VLAN component and the internal connections are achieved through the Customer Edge Port Configuration managed object defined here (12.13.3).

An internal connection between a CNP on the S-VLAN component and a PEP on the C-VLAN component is instantiated for each service instance. The CNP is identified by the CEP identifier and the S-VID value used for the PVID of the CNP. The PEP is identified by the CEP identifier and the C-VID value used for the PVID of the PEP. These PVID values and the connection between the CNP and PEP are established by reciprocal entries in the C-VID Registration Table (12.13.3.2) and the Provider Edge Port Configuration Table (12.13.3.4) as follows:

- a) An entry in the C-VID Registration Table is created for each C-VLAN supported in the C-VLAN component associated with a CEP. The CEP identifier and C-VID combination identify a PEP. The entry contains (among other parameters) an S-VID value corresponding to the PVID of the CNP, which associates the PEP with a specific CNP. Note that the CEP/C-VID combination identifies a single PEP, however multiple CEP/C-VID combinations can identify the same PEP if the entries for those CEP/C-VID combinations contain the same S-VID value. This many-to-one mapping of CEP/C-VID to PEP permits "bundling", i.e. mapping multiple C-VLANs to the same service instance.
- b) An entry in the Provider Edge Port Configuration Table is created for each S-VID corresponding to a service instance accessed by the C-VLAN component. The CEP identifier and S-VID combination identify a CNP. The entry contains (among other parameters) a C-VID value corresponding to the PVID of the PEP, which associates the CNP with a specific PEP. Note that the CEP/S-VID combination identifies a single CNP, however multiple CEP/S-VID combinations can identify the same CNP if the entries for those CEP/S-VID combinations contain the same C-VID value. This many-to-one mapping of CEP/S-VID to CNP permits configuration of asymmetric VLANs and can be used to establish rooted-multipoint connectivity (F.1.3.2).

Management control of the member sets and untagged sets for C-VIDs in the C-VLAN component is provided in C-VID Registration Table entries rather than through Static VLAN Registration Entries (thus eliminating the need for Filtering Database managed objects for the C-VLAN component). Management control of the member sets and untagged sets for S-VIDs in the S-VLAN component is provided through Static VLAN Registration Entries in the Filtering Database. Creating an entry for an S-VID in the Provider Edge Port Configuration table does not automatically modify the Static VLAN Registration Entries for the corresponding S-VLAN. A CNP is added to the member set and untagged set of an S-VLAN by including the CEP identifier (since the CEP identifier and S-VID combination identifies the CNP) in the Port Map of a Static VLAN Registration Entry for that S-VLAN (12.7.7.1, 8.8.2).

A C-VLAN component with more than one PEP (i.e., supporting more than one service instance) participates in the customer network Spanning Tree Protocol by running an instance of RSTP with the enhancements specified in 13.39. This protocol instance is managed using the managed objects defined in 12.8 and 12.12. All BPDUs generated by this protocol instance use the MAC address of the CEP as a source address and as the bridge address portion of Bridge Identifier. For each PEP, the protocol uses the S-VID value that is the PVID of the associated CNP as the port number. For the CEP, the protocol uses the value 0xFFFF as the port number.

The following managed objects define the semantics of the management operations specific to Provider Bridges:

- c) The Provider Bridge Port Type managed object (12.13.1);
- d) The Network Port Configuration managed object (12.13.2);
- e) The Customer Edge Port Configuration managed object (12.13.3).

12.13.1 Provider Bridge Port Type managed object

The management operations that can be performed on the Provider Bridge Port Type managed object are as follows:

- a) Read Provider Bridge Port Type (12.13.1.1);
- b) Configure Provider Bridge Port Type (12.13.1.2).

12.13.1.1 Read Provider Bridge Port Type

12.13.1.1.1 Purpose

To obtain information regarding the designated type of an externally accessible port on a Provider Bridge.

12.13.1.1.2 Inputs

- a) Port Number: the number of the Bridge Port.

12.13.1.1.3 Outputs

- a) Port Type: this takes one of the following values:
 - 1) Provider Network Port;
 - 2) Customer Network Port;
 - 3) Customer Edge Port.

12.13.1.2 Configure Provider Bridge Port Type

12.13.1.2.1 Purpose

To designate the type of an externally accessible port on a Provider Bridge.

12.13.1.2.2 Inputs

- a) Port Number: the number of the Bridge Port.
- b) Port Type: this takes one of the following values:
 - 1) Provider Network Port;
 - 2) Customer Network Port;
 - 3) Customer Edge Port.

12.13.1.2.3 Outputs

None.

12.13.2 Network Port Configuration managed object

The Network Port Configuration managed object applies to each externally accessible Customer Network Port or Provider Network Port on a Provider Bridge. It includes

- a) The VID Translation Table, which provides a bidirectional mapping between a local S-VID (used in data and protocol frames transmitted and received through this Customer Network Port or Provider Network Port) and a relay S-VID (used by the filtering and forwarding processes of the S-VLAN component in a Provider Bridge).

The management operations that can be performed are as follows:

- b) Read VID Translation Table Entry (12.13.2.1);
- c) Configure VID Translation Table Entry (12.13.2.2).

12.13.2.1 Read VID Translation Table Entry

12.13.2.1.1 Purpose

To read the relay S-VID associated with the local S-VID.

12.13.2.1.2 Inputs

- a) Port Number: the number of the Customer or Provider Network Port.
- b) Local VLAN Identifier: a 12-bit VID.

12.13.2.1.3 Outputs

- a) Relay VLAN Identifier: a 12-bit VID.

12.13.2.2 Configure VID Translation Table Entry

12.13.2.2.1 Purpose

To modify an entry in the VID Translation Table.

12.13.2.2.2 Inputs

- a) Port Number: the number of the Customer or Provider Network Port.
- b) Local VLAN Identifier: a 12-bit VID.
- c) Relay VLAN Identifier: a 12-bit VID.

12.13.2.2.3 Outputs

None.

12.13.3 Customer Edge Port Configuration managed object

The Customer Edge Port Configuration managed object applies to each externally accessible Customer Edge Port on a Provider Edge Bridge. It includes

- a) The C-VID Registration Table, which provides a mapping between a Customer VLAN Identifier (C-VID) and the service instance represented by a Service VLAN Identifier (S-VID) selected for that C-VLAN. This table provides the equivalent functionality of
 - 1) configuring the PVID of the internal Customer Network Port on the S-VLAN component;
 - 2) adding the corresponding Provider Edge Port on the C-VLAN component to the member set of the C-VLAN;
 - 3) adding the Provider Edge Port and/or Customer Edge Port to the untagged set of the C-VLAN (if it is desired that frames forwarded to that port are transmitted untagged for this C-VLAN).

- b) The Provider Edge Port configuration parameters, which provide the subset of the Bridge VLAN Configuration managed object (12.10.1) that is relevant for the internal ports of the C-VLAN component associated with the Customer Edge Port.
- c) The Service Priority Regeneration Table, which provides the Priority Regeneration Table (12.6.2) for each internal Customer Network Port connected to the C-VLAN component associated with the Customer Edge Port.

The management operations that can be performed are as follows:

- d) Read C-VID Registration Table Entry (12.13.3.1);
- e) Configure C-VID Registration Table Entry (12.13.3.2);
- f) Read Provider Edge Port Configuration (12.13.3.3);
- g) Set Provider Edge Port Configuration (12.13.3.4);
- h) Read Service Priority Regeneration Table (12.13.3.5);
- i) Set Service Priority Regeneration Table (12.13.3.6).

12.13.3.1 Read C-VID Registration Table Entry

12.13.3.1.1 Purpose

To read the VLAN Identifier of the service associated with a specific Customer VLAN in the C-VLAN component of a Provider Edge Bridge.

12.13.3.1.2 Inputs

- a) Port Number: the number of the Customer Edge Port;
- b) Customer VLAN Identifier: a 12-bit C-VID.

12.13.3.1.3 Outputs

- a) Service VLAN Identifier: a 12-bit S-VID;
- b) Untagged PEP: a boolean indicating frames for this C-VLAN should be forwarded untagged through the Provider Edge Port;
- c) Untagged CEP: a boolean indicating frames for this C-VLAN should be forwarded untagged through the Customer Edge Port.

12.13.3.2 Configure C-VID Registration Table Entry

12.13.3.2.1 Purpose

To modify an entry in the C-VID Registration Table.

12.13.3.2.2 Inputs

- a) Port Number: the number of the Customer Edge Port;
- b) Customer VLAN Identifier: a 12-bit C-VID;
- c) Service VLAN Identifier: a 12-bit S-VID;
- d) Untagged PEP: a boolean indicating frames for this C-VLAN should be forwarded untagged through the Provider Edge Port;
- e) Untagged CEP: a boolean indicating frames for this C-VLAN should be forwarded untagged through the Customer Edge Port.

12.13.3.2.3 Outputs

None.

12.13.3.3 Read Provider Edge Port Configuration

12.13.3.3.1 Purpose

To read the current configuration of an internal Provider Edge Port in the C-VLAN component of a Provider Edge Bridge.

12.13.3.3.2 Inputs

- a) Port Number: the number of the Customer Edge Port;
- b) Service VLAN Identifier: a 12-bit S-VID.

12.13.3.3.3 Outputs

- a) PVID: a 12-bit C-VID to be used for untagged frames received at the Provider Edge Port;
- b) Default User Priority: an integer range 0-7 to be used for untagged frames received at the Provider Edge Port;
- c) Acceptable Frame Types: the Acceptable Frame Types (8.6.2) for frames received at the Provider Edge Port. The permissible values for the parameter are as follows:
 - 1) *Admit only VLAN-Tagged frames*;
 - 2) *Admit only Untagged and Priority-Tagged frames*;
 - 3) *Admit all frames*;
- d) Enable Ingress Filtering: the Enable Ingress Filtering parameter for frames received at the Provider Edge Port. The permissible values for the parameter are as follows:
 - 1) Enabled;
 - 2) Disabled.

12.13.3.4 Set Provider Edge Port Configuration

12.13.3.4.1 Purpose

To modify the configuration of a Provider Edge Port in the C-VLAN component of a Provider Edge Bridge.

12.13.3.4.2 Inputs

- a) Port Number: the number of the Customer Edge Port;
- b) Service VLAN Identifier: a 12-bit S-VID;
- c) PVID: a 12-bit C-VID to be used for untagged frames received at the Provider Edge Port;
- d) Default User Priority: an integer range 0-7 to be used for untagged frames received at the Provider Edge Port;
- e) Acceptable Frame Types: the Acceptable Frame Types (8.6.2) for frames received at the Provider Edge Port. The permissible values for the parameter are as follows:
 - 1) *Admit only VLAN-Tagged frames*;
 - 2) *Admit only Untagged and Priority-Tagged frames*;
 - 3) *Admit all frames*;
- f) Enable Ingress Filtering: the Enable Ingress Filtering parameter for frames received at the Provider Edge Port. The permissible values for the parameter are as follows:
 - 1) Enabled;
 - 2) Disabled.

12.13.3.4.3 Outputs

None.

12.13.3.5 Read Service Priority Regeneration Table

12.13.3.5.1 Purpose

To read the current contents of the Priority Regeneration Table (6.9.3) for an internal Customer Network Port connected to the C-VLAN component associated with a Customer Edge Port.

12.13.3.5.2 Inputs

- a) Port Number: the number of the Customer Edge Port;
- b) Service VLAN Identifier: a 12-bit S-VID.

12.13.3.5.3 Outputs

- a) Regenerated Priority value for Received Priority 0: Integer in range 0–7;
- b) Regenerated Priority value for Received Priority 1: Integer in range 0–7;
- c) Regenerated Priority value for Received Priority 2: Integer in range 0–7;
- d) Regenerated Priority value for Received Priority 3: Integer in range 0–7;
- e) Regenerated Priority value for Received Priority 4: Integer in range 0–7;
- f) Regenerated Priority value for Received Priority 5: Integer in range 0–7;
- g) Regenerated Priority value for Received Priority 6: Integer in range 0–7;
- h) Regenerated Priority value for Received Priority 7: Integer in range 0–7.

12.13.3.6 Set Service Priority Regeneration Table

12.13.3.6.1 Purpose

To modify the contents of the Priority Regeneration Table (6.9.3) for an internal Customer Network Port connected to the C-VLAN component associated with a Customer Edge Port.

12.13.3.6.2 Inputs

- a) Port Number: the number of the Customer Edge Port;
- b) Service VLAN Identifier: a 12-bit S-VID;
- c) Regenerated Priority value for Received Priority 0: Integer in range 0–7;
- d) Regenerated Priority value for Received Priority 1: Integer in range 0–7;
- e) Regenerated Priority value for Received Priority 2: Integer in range 0–7;
- f) Regenerated Priority value for Received Priority 3: Integer in range 0–7;
- g) Regenerated Priority value for Received Priority 4: Integer in range 0–7;
- h) Regenerated Priority value for Received Priority 5: Integer in range 0–7;
- i) Regenerated Priority value for Received Priority 6: Integer in range 0–7;
- j) Regenerated Priority value for Received Priority 7: Integer in range 0–7.

12.13.3.6.3 Outputs

None.

12.14 CFM entities

The CFM managed objects model operations that modify, or inquire 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 17.4.9 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 17.3.9 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.

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).

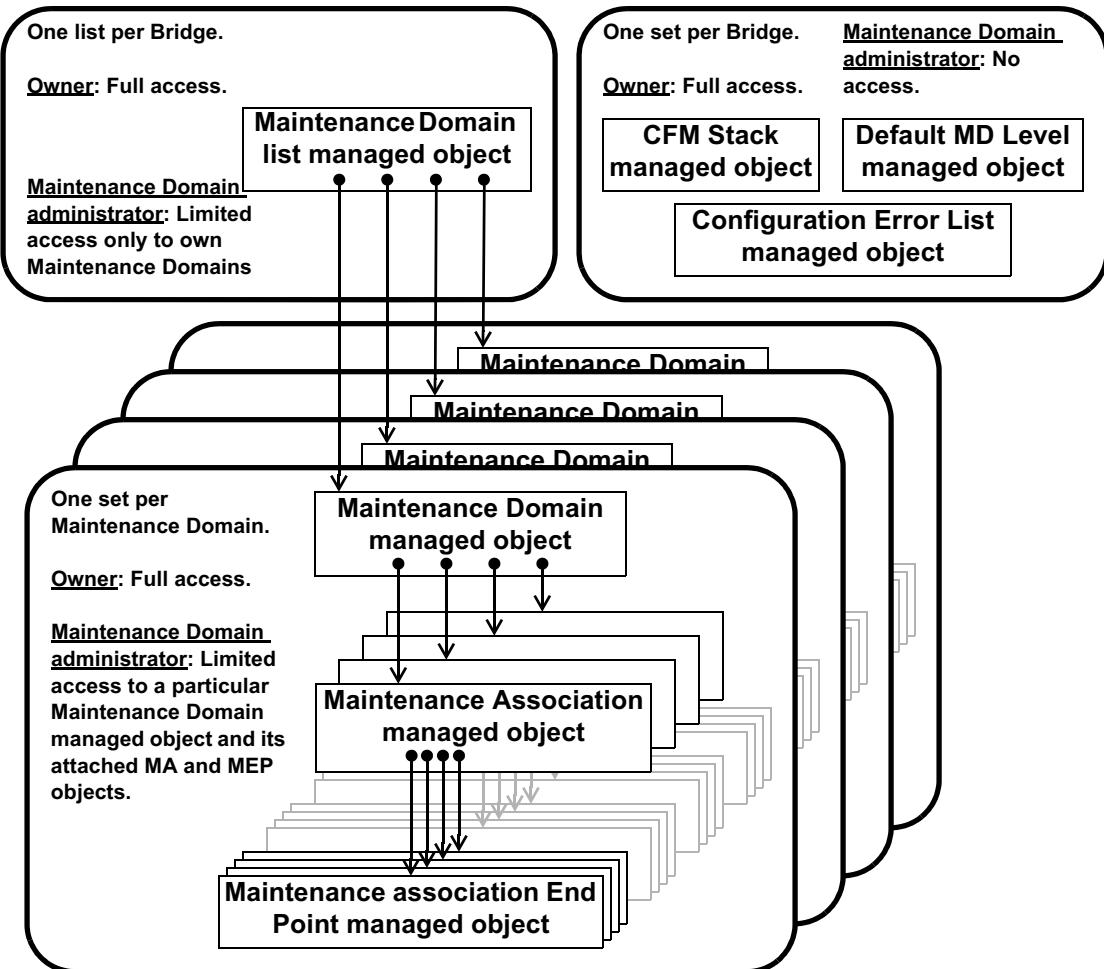


Figure 12-1—Relationships among CFM managed objects

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 or I-SID for I-components or B-components or the TE-SID to which the MEPs and MHFs are attached, or 0, for those attached to no VID or I-SID or TE-SID.

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 component. It controls MIP Half Function (MHF) creation for VIDs or I-SIDs of I- or B-components 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 not associated with any MA configured on the Bridge component.

12.14.3.1.2 Input

- a) A VID or I-SID in an I- or B-component.

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, or the Backbone-SID of the B-component or VIP-SID of the I-component associated with any MHF on the I-SID named in item a) in 12.14.3.1.2. 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 or I-SID of I- or B-components 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) or I-SID(s) of I- or B-components: defMHFnone: (n, the default), defMHFdefault (o), or defMHFexplicit (p) (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: 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 or on I-SIDs used in an I- or B-component not associated with any MA.

12.14.3.2.2 Inputs

- a) A list of VIDs or the VIP-SIDs used in the I-component or Backbone-SID used in the B-component, or 0, indicating no VID or I-SID. 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 or I-SIDs to this MD Level would exceed the number of MD Levels supported by some Bridge Port (item c) in);
 - 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 is a list of {service instance identifier, port} pairs configured in error together with the identity of the configuration error. The service instance identifier is a VID, I-SID, or TE-SID and the port is a simple bridge port or aggregated bridge port.

12.14.4.1 Read Configuration Error List managed object

12.14.4.1.1 Purpose

To list the entries in the Configuration Error List where each entry describes the Service Instance Identifier and port pair that identifies the entry, together with a description of the particular configuration error.

12.14.4.1.2 Inputs

- a) A vlan_identifier or I-SID or TE-SID, specifying which service instance to check for 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:
 - 1) Operation rejected due to illegal vlan_identifier;
 - 2) Operation rejected due to illegal or nonexistent 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: defMHFnone: (n, the default value), defMHFdefault (o), or defMHFexplicit (p) (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: 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). In the case of a PBB-TE or a VID-based MA, the 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, I-SID, or the TE-SID monitored by this MA, or 0, if the MA is not attached to a VID, or I-SID, or TE-SID. In the case of a list of VIDs, the first VID in the list is the MA's Primary VID (default none). The specification of I-SID is allowed only in the case of I- or B- components. The TE-SID is allowed only in the case that PBB-TE is supported.

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 rejected because the I-SID value specified does not correspond to a valid Backbone Service Instance;
 - 7) Operation rejected because the TE-SID value specified does not correspond to a valid TESI;
 - 8) 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), I-SID, or TE-SID monitored by this MA, or 0, if the MA is not attached to any VID, I-SID, or TE-SID. In the case of a list of VIDs, 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: defMHFnone: (n), defMHFdefault (o), defMHFexplicit (p), or defMHFdefer (q), (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:
 - 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), I-SID, or TE-SID as this MEP (or for no VID, I-SID, or TE-SID , if this MEP's MA has no VID, I-SID, or TE-SID), on that Bridge Port or aggregated IEEE 802.3 port;
 - 5) Operation failed because one or more of the listed VIDs, I-SID, or TE-SID 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, I-SID, or TE-SID;
 - 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 either that the Primary VID is that of the MEP's MA or that the MEP's MA is associated with no VID. In the case of a PBB-TE associated MEP, the Primary VID is not writable but is always associated with the value of the ESP-VID parameter identifying the MA's ESP that has the MEP's MAC address in its ESP-SA field (MEPprimaryVID, 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 (20.37);
- g) (writable) A Boolean flag indicating whether the MEP is or is not to generate CCMs (CClenabled, 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, I-SID, or TE-SID);

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). In the case of a MEP that is associated with a TESI, the MAC address of the CBP upon which the MEP is operating;
- 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):
 - 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 (20.35.3) (default 2.5 s);
- m) (writable) The time that defects must be absent before resetting a Fault Alarm (20.35.4) (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 (20.35.9):
 - 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 (20.35.7) indicating that some other MEP in this MEP's MA is transmitting the RDI bit (can trigger DefRDICCM);
- p) A Boolean flag (20.35.6) 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 (20.35.5) 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 (20.30.2);
- y) The total number of valid, in-order LBRs received (20.33.1);
- z) The total number of valid, out-of-order LBRs received (20.33.1);
- 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 (20.41.1);
- ac) The total number of unexpected LTRs received (20.44.1);
- ad) The total number of LBRs transmitted (20.28.2);
- ae) (writable) A list indicating which of the remote MEPs in the same MA are active. The Remote MEP state machines (20.20) are instantiated only for the active remote MEPs. By default, all configured remote MEPs in the same MA are active;
- af) (writable) Only applicable for PBB-TE MEPs supporting the Traffic field (21.6.1.4), a Boolean flag indicating that the capability to report the presence of traffic is enabled. The default value is FALSE;
- ag) (writable) Only applicable for PBB-TE MEPs supporting the Traffic field (21.6.1.4), a Boolean flag indicating that the mismatch defect is allowed to generate a Fault Alarm. The default value is FALSE;
- ah) Only applicable for PBB-TE MEPs supporting the Traffic field (21.6.1.4), a Boolean flag indicating the presence of a traffic field mismatch defect (20.25.2) (can trigger DefMmCCM);

- ai) Only applicable for PBB-TE MEPs supporting the Traffic field (21.6.1.4), a Boolean flag indicating the presence of a local mismatch defect (20.25.5) (can trigger DefMmCCM);
- aj) Only applicable for PBB-TE MEPs supporting the Traffic field (21.6.1.4), an enumerated value indicating if the mismatch defect that has been present since the MEP Mismatch Fault Notification Generator state machine (20.40) was last in the MFNG_RESET state, either:
 - 1) **DefNone:** No defect has been present since the last MFNG_RESET state; or
 - 2) **DefMmCCM:** This MEP is receiving CCMs that have different value in their Traffic field than the CCMs transmitted by this MEP or is transmitting CCMs with both the RDI and the Traffic field bit set to 1 and the disableLocdefect (20.25.4) variable is cleared, so variable item ah) in 12.14.7.1.3 is set;
- ak) Only applicable for PBB-TE MEPs supporting the Traffic field (21.6.1.4), a value indicating the current state of the MEP Mismatch Fault Notification Generator state machine (20.40).

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 (20.31.1):
 - 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 (20.31.1) (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].

- f) In the case of a PBB-TE MEP, the Reverse VID to be used in the associated PBB-TE MIP TLV (21.7.5). The parameter is not required in the case of PBB-TE MEPs associated with point-to-point TESIs or if the b2) entry is a MAC address corresponding to an entry in the ESP-DA field of any of the MA's ESPs.

12.14.7.3.3 Outputs

- a) An indication of whether the command was accepted by the MEP:
 - 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) Operation rejected due to invalid ESP-VID;
 - 6) The LBM(s) will be (or have been) sent; or
 - 7) 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 (20.30.2) 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 (20.42.1);
- 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;
- d) An initial value for the LTM TTL field (21.8.4). Default value, if not specified, is 64;
- e) In the case of a PBB-TE MEP, the Reverse VID to be used in the associated PBB-TE MIP TLV (21.7.5). The parameter is not required in the case of PBB-TE MEPs associated with point-to-point TESIs.

12.14.7.4.3 Outputs

- a) An indication of whether the command was accepted by the MEP:
 - 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) Operation rejected due to invalid ESP-VID;
 - 6) The command was accepted and the LTM has (or will be) transmitted; or
 - 7) 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 (20.41.1) 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 (20.41.2) 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 (20.41.2). Each LTR entry in the list returns:

- a) An indication of whether the command was accepted by the MEP:
 - 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 (20.41.2.2);
- c) A Boolean value stating whether an LTM was forwarded by the responding MP, as returned in the FwdYes flag of the Flags field (20.41.2.1);
- 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 (20.41.2.1);
- e) An octet string holding the Last Egress Identifier field returned in the LTR Egress Identifier TLV of the LTR (20.41.2.3);
- f) An octet string holding the Next Egress Identifier field returned in the LTR Egress Identifier TLV of the LTR (20.41.2.4);
- g) An enumerated value indicating the value returned in the Relay Action field (20.41.2.5): 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 (20.41.2.6) of the LTR: IngOK, IngDown, IngBlocked, or IngVID;
- l) (if returned in the LTR) The MAC address returned in the Ingress MAC Address field (20.41.2.7);
- m) (if returned in the LTR) An enumerated value indicating the format of the Ingress Port ID field (20.41.2.8);

- n) (if returned in the LTR) The Ingress Port ID (20.41.2.9);
- o) (if returned in the LTR) An enumerated value indicating the value returned in the Egress Action field (20.41.2.10) of the LTR: EgrOK, EgrDown, EgrBlocked, or EgrVID;
- p) (if returned in the LTR) The MAC address returned in the Egress MAC Address field (20.41.2.11);
- q) (if returned in the LTR) An enumerated value indicating the format of the Egress Port ID (20.41.2.12);
- r) (if returned in the LTR) The Egress Port ID (20.41.2.13); 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:
 - 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:
 - 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:
 - 1) **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:
 - 1) **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:
 - 1) **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 (20.37), or by the MEP Mismatch Fault Notification Generator state machine (20.40). 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 (20.35.9), and to the corresponding MEP managed object [item n) in 12.14.7.1.3] or indicating the presence of a mismatch defect [item aj) in 12.14.7.1.3].

12.15 Backbone Core Bridge management

Each Backbone Core Bridge is a conformant S-VLAN Bridge (5.10.1) that is managed by the managed objects summarized in 12.2 and specified in detail in 12.4 through 12.14.

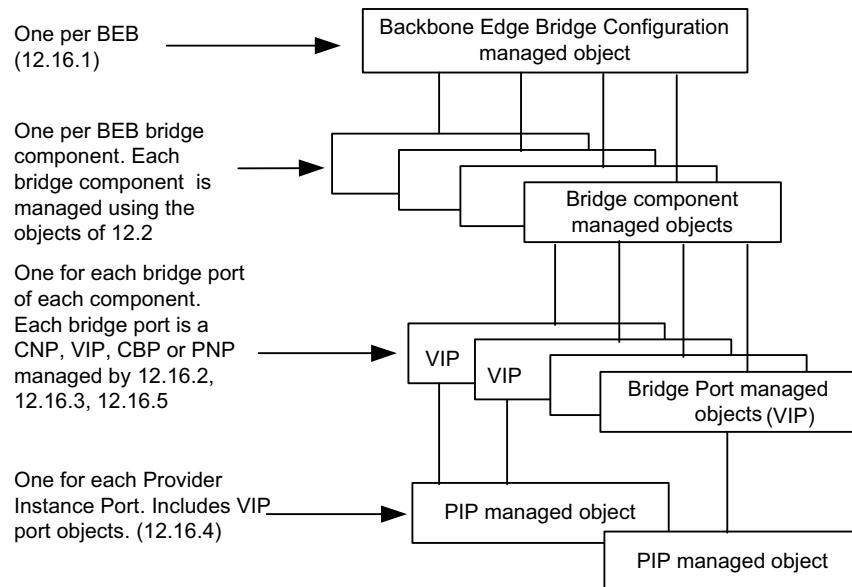
12.16 Backbone Edge Bridge management

The conformance requirements for Backbone Edge Bridges (BEBs) are specified in 5.11. Each of the I-components and (or) the B-component of a BEB are managed using the managed objects defined in 12.2, while this subclause adds managed objects used in conjunction with those of 12.2 to manage the overall configuration and the VIPs, PIPs, and CBPs of a BEB. This subclause specifies the additional managed objects specific to the operation of BEBs, which

- a) Support a BEB configuration managed object identifying all the BEB components (12.16.1);
- b) Identify BEB port types (12.16.2);
- c) Provide additional managed objects for management of the Virtual Instance Ports (VIPs), Provider Instance Ports (PIP) and Customer Backbone Ports (CBPs) (12.16.3, 12.16.4, 12.16.5).

Only a single view of the BEB managed objects exists. This view may only contain the objects needed for the type of BEB under management. If the BEB contains only a single I-component then the view will have only the BEB configuration, BEB port configuration, VIP configuration, and PIP configuration managed objects. If the BEB has only a B-Comp then the view will have only the BEB configuration, BEB port configuration, and CBP configuration managed objects.

An instance of a managed object associated with a BEB is referenced by a unique identifier. Rules associated with the selection of this identifier are specified within this subclause. Each BEB has a single BEB configuration managed object, which is described in this subclause and illustrated in Figure 12-2. This managed object is accessed at only one of the components within a given BEB, the B-component if that is present, and the single I-component otherwise, and is referenced by that component's MAC address.

**Figure 12-2—Relationship among BEB managed objects**

Each VLAN-aware Bridge component relays frames between Bridge Ports. In a BEB, these Bridge Ports can provide external connectivity directly, or can connect to other bridge components through internal LANs. Each component contained within a BEB is associated with the BEB configuration managed object representing that BEB. The component is uniquely identified within the PBBN by the combination of a componentID unique within the BEB and by the MAC address of the BEB within that the component is contained. Each component may also be identified by the MAC address of the component that is unique within the PBBN. Within a BEB, a B-component uses two types of Bridge Ports—PNPs and CBPs—and an I-component uses two types of Bridge Ports—Virtual Instance Ports and CNPs. Each Bridge Port is represented by a Bridge Port managed object (12.4.2), a Provider Bridge Port Type managed object (12.13.1), and a BEB port configuration managed object (12.16.2). Each such managed object is uniquely identified, within the BEB, by the combination of the ComponentID and Port Number (12.3) of the associated Bridge Port.

Each VIP associated with an I-component is mapped to a PIP associated with that I-component. In general, multiple VIPs are associated with a single PIP. PIPs are not Bridge Ports, though they are ports on BEBs, as they are not used directly by the MAC Relay Entity (see Clause 8) of a bridge component. Each VIP is supported by one and only one PIP. Each PIP is therefore identified uniquely by the ComponentID and Port Number of any of the VIPs that it supports and by convention is referenced by the smallest Port Number supported by the PIP. Each PIP is managed using the PIP configuration managed object (12.16.4), and each VIP is managed using the VIP configuration managed object of (12.16.3).

PIPs and CBPs of a BEB may be internal or external. Each externally accessible port on a BEB is designated as a PNP, CBP, PIP, or CNP.

The following managed objects define the semantics of the management operations specific to BEBs:

- d) The BEB configuration managed object (12.16.1)
- e) The BEB/PB/VLAN Bridge Port configuration managed object (12.16.2)
- f) The VIP configuration managed object (12.16.3)
- g) The PIP configuration managed object (12.16.4)
- h) The CBP configuration managed object (12.16.5)

NOTE—All objects specified in 12.16 are persistent over power-up/down and system reboot unless otherwise specified.

12.16.1 BEB configuration managed object

The BEB configuration managed object identifies a BEB and all the managed objects that manage the BEB. The management operations that can be performed on the BEB configuration managed object are as follows:

- a) Read BEB configuration (12.16.1.1)
- b) Set BEB configuration (12.16.1.2)
- c) Create BEB component (12.16.1.3)
- d) Delete BEB component (12.16.1.4)
- e) Create BEB Bridge Port (12.16.1.5)
- f) Create BEB PIP (12.16.1.6)
- g) Delete BEB Bridge Port (12.16.1.7)
- h) Delete BEB PIP (12.16.1.8)

12.16.1.1 Read BEB configuration

12.16.1.1.1 Purpose

All BEBs shall implement the read BEB configuration function to obtain information regarding the type of components and ports within a BEB.

12.16.1.1.2 Inputs

None.

12.16.1.1.3 Outputs

- a) BEB address—the MAC address for the BEB used to access the BEB configuration managed object (8.13.8). Each BEB has a single BEB address even if the BEB has many components.
- b) BEB name—a text string of up to 32 characters, of locally determined significance.
- c) Number of I-components—in the BEB, which may range from 0 to many.
- d) I-component addresses—a list specifying the following for each I-component or a NULL list value for no I-components:
 - 1) ComponentID—the number of the I-component.
 - 2) A Bridge address that is the MAC address for the I-Component (8.13.8).
- e) Number of B-components—in the BEB, which may be 0 or 1.
- f) B-component address—a list specifying the following for the B-component or a NULL list value for no B-component.
 - 1) ComponentID—the number of the B-component.
 - 2) A Bridge address that is the MAC address for the B-component (8.13.8).
- g) Number of BEB ports (MAC Entities).
- h) Port addresses—a list specifying the following for each port:
 - 1) ComponentID—the number identifying the bridge component associated with this port.
 - 2) An interface—either that component's Port Number for the Bridge Port, or in the case of a PIP, the PIP Index.
 - 3) Port address—the specific MAC address associated with the port (8.13.2).
 - 4) External port—a Boolean indicating that the port is an external port.

12.16.1.2 Set BEB configuration

12.16.1.2.1 Purpose

All BEBs shall implement the set BEB configuration function to set the BEB Name. The BEB Name is persistent over power-up or reboot.

12.16.1.2.2 Inputs

- a) BEB Name—a text string of up to 32 characters, of locally determined significance.

12.16.1.2.3 Outputs

None.

12.16.1.3 Create BEB component (optional)

12.16.1.3.1 Purpose

A BEB may implement the create BEB component function to allow the dynamic creation of new components. Components created by this command are persistent over power-up or reboot.

12.16.1.3.2 Inputs

- a) ComponentType: this takes one of the following values:
 - 1) I-Comp
 - 2) B-Comp

12.16.1.3.3 Outputs

None.

12.16.1.4 Delete BEB component (optional)

12.16.1.4.1 Purpose

A BEB may implement the delete BEB component function to allow deletion of components. Component deletion using this command is persistent over power-up or reboot.

12.16.1.4.2 Inputs

- a) ComponentID—indicates the component being deleted.

12.16.1.4.3 Outputs

None.

12.16.1.5 Create BEB Bridge Port (optional)

12.16.1.5.1 Purpose

A BEB may implement the create BEB Bridge Port function to allow dynamic creation of Bridge Ports. Bridge Ports created by this command are persistent over power-up or reboot.

12.16.1.5.2 Inputs

- a) Port Index—indicates the port where the new Bridge Port will be created.
- b) ComponentID—indicates the component where the new Bridge Port will be created.
- c) Port type—this takes one of the following values:
 - 1) CNP
 - 2) PNP
 - 3) CBP
 - 4) VIP

12.16.1.5.3 Outputs

None.

12.16.1.6 Create BEB PIP (optional)

12.16.1.6.1 Purpose

A BEB may implement the create BEB PIP function to allow dynamic creation of PIPs. PIPs created by this command are persistent over power-up or reboot.

12.16.1.6.2 Inputs

- a) Port Index—indicates the port where the new PIP will be created.
- b) ComponentID—indicates the component where the new PIP will be created.

12.16.1.6.3 Outputs

None.

12.16.1.7 Delete BEB Bridge Port (optional)

12.16.1.7.1 Purpose

A BEB may implement the delete BEB Bridge Port function to allow deletion of Bridge Ports. Bridge Port deletion using this command is persistent over power-up or reboot.

12.16.1.7.2 Inputs

- a) ComponentID—the component for the Bridge Port.
- b) Port Number—the Port Number for the Bridge Port.

12.16.1.7.3 Outputs

None.

12.16.1.8 Delete BEB PIP (optional)

12.16.1.8.1 Purpose

A BEB may implement the delete BEB PIP function to allow deletion of PIPs. PIP deletion using this command is persistent over power-up or reboot.

12.16.1.8.2 Inputs

- a) PIP Index—identifies the PIP within the BEB.

12.16.1.8.3 Outputs

None.

12.16.2 BEB/PB/VLAN Bridge Port configuration managed object

The management operations that can be performed on the BEB/PB/VLAN Bridge Port configuration managed object are as follows:

- a) Read BEB/PB/VLAN Bridge port configuration (12.16.2.1).

12.16.2.1 Read BEB/PB/VLAN Bridge port configuration

12.16.2.1.1 Purpose

All BEBs shall implement the read BEB/PB/VLAN Bridge port configuration function to obtain information regarding the port type of a Bridge Port on the BEB.

12.16.2.1.2 Inputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component’s Port Number for the Bridge Port.

12.16.2.1.3 Outputs

- a) Port Type—this takes one of the following values:
 - 1) C-VLAN Port
 - 2) PNP (identical to the PNP Bridge Port type specified by 12.13.1)
 - 3) CNP (identical to the CNP Bridge Port type specified by 12.13.1)
 - 4) CEP (identical to the CEP Bridge Port type specified by 12.13.1)
 - 5) CBP (5.11, 6.11, 12.13.1 will indicate PNP)
 - 6) VIP (5.11, 6.10, 12.13.1 will indicate PNP)

NOTE—Support for C-VLAN ports requires equipment with VLAN Bridge (5.9) functions in addition to BEB (5.11) functions. Support for CEP ports requires equipment with PB (5.10) functions in addition to BEB functions.

12.16.3 VIP configuration managed object

The VIP configuration managed object applies to each VIP (6.10) on a BEB whether its associated PIP is externally or internally accessible.

It includes

- a) The VIP configuration parameters, which provide the subset of the Bridge VLAN Configuration managed object (12.10.1) that is relevant for the configuration of the backbone service instance.

The management operations that can be performed are as follows:

- b) Read VIP configuration (12.16.3.1).
- c) Set VIP configuration (12.16.3.2).

12.16.3.1 Read VIP configuration

12.16.3.1.1 Purpose

All BEBs with I-components shall implement the read VIP configuration function to allow reading the current configuration of the VIPs (6.10) in an I-component of a BEB. Each VIP is configured by 12.8, 12.10, 12.16.3.2. The adminPointToPointMAC parameter is configured by 12.8.2.3.2 while the PVID and Acceptable Frame Types parameters are supplied by the Bridge Port configuration of 12.10.1.2 and 12.10.1.3.

12.16.3.1.2 Inputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component's Port Number for the Bridge Port.

12.16.3.1.3 Outputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component's Port Number for the Bridge Port.
- c) PIP Name—a text string of up to 32 characters that identifies the PIP within a BEB with which this VIP is associated.
- d) VIP-ISID—the I-SID associated with this VIP (6.10).
- e) Default Backbone Destination—an EUI-48 DA (6.10.2).
- f) Ingress, egress, both ingress and egress—a 2-bit selector that determines if frames on this VIP may ingress to the PBBN but not egress the PBBN, egress to the PBBN but not ingress the PBBN, or both ingress and egress the PBBN. This feature is used to support asymmetric VLANs (F.1.3).
- g) enableConnectionIdentifier—A Boolean to indicate if the connection_identifier parameter is allowed to learn associations between a backbone MAC address and a customer MAC address (6.10).

12.16.3.2 Set VIP configuration

12.16.3.2.1 Purpose

All BEBs with I-components shall implement the set VIP configuration function to allow setting the configuration of the internal VIP in the I-component of a BEB.

12.16.3.2.2 Inputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component's Port Number for the Bridge Port.
- c) VIP-ISID—the I-SID associated with this VIP (6.10).
- d) Ingress, egress, both ingress and egress—a 2-bit selector that determines if frames on this VIP may ingress to the PBBN but not egress the PBBN, egress to the PBBN but not ingress the PBBN, or both ingress and egress the PBBN. This feature is used to support asymmetric VLANs (F.1.3).
- e) enableConnectionIdentifier—A Boolean that determines if the connection_identifier parameter is allowed to learn associations between a backbone MAC address and a customer MAC address (6.10). The default value is TRUE. This parameter should be configured to FALSE at the root node of a point-to-multipoint TESI.

12.16.3.2.3 Outputs

- a) Operation status. This takes one of the following values:

- 1) Operation rejected because the Bridge Port identified by the ComponentID and Port Number is not a VIP;
- 2) Operation accepted.

12.16.4 PIP configuration managed object

The PIP Configuration managed object applies to each PIP (6.10) on a BEB.

It includes the following:

- a) The VIP mapping, which provides the set of VIPs multiplexed on this PIP
- b) The PIP MAC address, which is used as the B-SA for all encapsulated frames

The management operations that can be performed are as follows:

- c) Read PIP Configuration (12.16.4.1)
- d) Optionally, Set PIP Configuration (12.16.4.2)
- e) Read VIP to PIP mapping (12.16.4.3)
- f) Optionally, Set VIP to PIP mapping (12.16.4.4)
- g) Read PIP Priority Code Point Selection (12.16.4.5)
- h) Set PIP Priority Code Point Selection (12.16.4.6)
- i) Read PIP Priority Code Point Decoding Table (12.16.4.7)
- j) Optionally, Set PIP Priority Code Point Decoding Table (12.16.4.8)
- k) Read PIP Priority Code Point Encoding Table (12.16.4.9)
- l) Optionally, Set PIP Priority Code Point Encoding Table (12.16.4.10)
- m) Read PIP Use_DEI parameter (12.16.4.11)
- n) Optionally, Set PIP Use_DEI parameter (12.16.4.12)
- o) Read PIP Require Drop Encoding parameter (12.16.4.13)
- p) Optionally, Set PIP Require Drop Encoding parameter (12.16.4.14)

12.16.4.1 Read PIP configuration

12.16.4.1.1 Purpose

All BEBs with I-components shall implement the read PIP configuration function to allow reading the current configuration of the PIP in the I-component of a BEB.

12.16.4.1.2 Inputs

- a) PIP Index—an identifier for the PIP within this BEB.

12.16.4.1.3 Outputs

- a) PIP B-MAC address—identifies the PIP (6.10.2).
- b) PIP Name—a text string of up to 32 characters used to identify a PIP within a BEB.
- c) ComponentID—the number identifying the I-component with this PIP.
- d) VIP map—a bit array of 4094 bits indicating all the VIPs associated with this port.

12.16.4.2 Set PIP configuration (optional)

12.16.4.2.1 Purpose

All BEBs with I-components may implement the set PIP configuration function to set the current configuration of the PIP in the I-component of a BEB.

12.16.4.2.2 Inputs

- a) PIP Index—an identifier for the PIP within this BEB.
- b) PIP B-MAC address—the B-MAC address used by this PIP in the B-SA field of transmitted frames.
- c) PIP Name—a text string of up to 32 characters used to identify the PIP within a BEB.
- d) VIP map—a bit array of 4094 bits indicating all the VIPs associated with this PIP.

12.16.4.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because the PIP Index is invalid.
 - 2) Operation rejected because the VIP map identifies a port that is not a VIP.
 - 3) Operation accepted.

12.16.4.3 Read VIP to PIP mapping

12.16.4.3.1 Purpose

All BEBs with I-components shall implement the read VIP to PIP mapping function to allow reading the VIP to PIP map information from the PIP configuration managed object.

12.16.4.3.2 Inputs

- a) ComponentID—the number identifying a bridge component with a VIP.
- b) Port Number—that component's Port Number for this VIP.

12.16.4.3.3 Outputs

- a) ComponentID—the number identifying a bridge component with a VIP.
- b) Port Number—that component's Port Number for this VIP.
- c) PIP Index—identifies the PIP supporting this VIP.
- d) PIP B-MAC address—the B-MAC address used by this PIP supporting this VIP (6.10.2).
- e) PIP Name—a text string of up to 32 characters identifying the PIP supporting this VIP.

12.16.4.4 Set VIP to PIP mapping (optional)

12.16.4.4.1 Purpose

All BEBs with I-components may implement the set VIP to PIP mapping function to allow binding individual VIPs to PIPs. The set VIP to PIP mapping function allows setting map entries for individual VIPs while the set PIP configuration function sets the entire VIP to PIP map.

12.16.4.4.2 Inputs

- a) ComponentID—the number identifying a bridge component with a VIP.
- b) Port Number—that component's Port Number for this VIP.
- c) PIP Index—an identifier for this PIP within this component of the BEB.

12.16.4.4.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because the ComponentID and Port Number do not identify a VIP.
 - 2) Operation rejected because the PIP Index does not identify a PIP within the specified component.

- 3) Operation accepted.

12.16.4.5 Read PIP Priority Code Point Selection

12.16.4.5.1 Purpose

To read which row of the Priority Code Point Encoding Table and Priority Code Point Decoding Table (6.10.3) is currently selected for use on this PIP.

12.16.4.5.2 Inputs

- a) PIP Index—an identifier for this PIP within this component of the BEB.

12.16.4.5.3 Outputs

- a) Priority Code Point Selection: this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D

12.16.4.6 Set PIP Priority Code Point Selection

12.16.4.6.1 Purpose

To set which row of the Priority Code Point Encoding Table and Priority Code Point Decoding Table (6.10.3) will be selected for use on this PIP.

12.16.4.6.2 Inputs

- a) PIP Index—an identifier for this PIP within this component of the BEB.
- b) Priority Code Point Selection—this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D

12.16.4.6.3 Outputs

None.

12.16.4.7 Read PIP Priority Code Point Decoding Table

12.16.4.7.1 Purpose

To read the current contents of a row in the Priority Code Point Decoding Table (6.10.3) for a PIP.

12.16.4.7.2 Inputs

- a) PIP Index—an identifier for this PIP.
- b) Priority Code Point Row—this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D

4) 5P3D

12.16.4.7.3 Outputs

- a) Priority value for Priority Code Point 0: Integer in range 0–7.
- b) Drop_eligible value for Priority Code Point 0: Boolean.
- c) Priority value for Priority Code Point 1: Integer in range 0–7.
- d) Drop_eligible value for Priority Code Point 1: Boolean.
- e) Priority value for Priority Code Point 2: Integer in range 0–7.
- f) Drop_eligible value for Priority Code Point 2: Boolean.
- g) Priority value for Priority Code Point 3: Integer in range 0–7.
- h) Drop_eligible value for Priority Code Point 3: Boolean.
- i) Priority value for Priority Code Point 4: Integer in range 0–7.
- j) Drop_eligible value for Priority Code Point 4: Boolean.
- k) Priority value for Priority Code Point 5: Integer in range 0–7.
- l) Drop_eligible value for Priority Code Point 5: Boolean.
- m) Priority value for Priority Code Point 6: Integer in range 0–7.
- n) Drop_eligible value for Priority Code Point 6: Boolean.
- o) Priority value for Priority Code Point 7: Integer in range 0–7.
- p) Drop_eligible value for Priority Code Point 7: Boolean.

12.16.4.8 Set PIP Priority Code Point Decoding Table (optional)

12.16.4.8.1 Purpose

To modify the contents of a row in the Priority Code Point Decoding Table (6.10.3) for a PIP.

12.16.4.8.2 Inputs

- a) PIP Index—an identifier for this PIP.
- b) Priority Code Point Row—this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D
- c) Priority value for Priority Code Point 0: Integer in range 0–7.
- d) Drop_eligible value for Priority Code Point 0: Boolean.
- e) Priority value for Priority Code Point 1: Integer in range 0–7.
- f) Drop_eligible value for Priority Code Point 1: Boolean.
- g) Priority value for Priority Code Point 2: Integer in range 0–7.
- h) Drop_eligible value for Priority Code Point 2: Boolean.
- i) Priority value for Priority Code Point 3: Integer in range 0–7.
- j) Drop_eligible value for Priority Code Point 3: Boolean.
- k) Priority value for Priority Code Point 4: Integer in range 0–7.
- l) Drop_eligible value for Priority Code Point 4: Boolean.
- m) Priority value for Priority Code Point 5: Integer in range 0–7.
- n) Drop_eligible value for Priority Code Point 5: Boolean.
- o) Priority value for Priority Code Point 6: Integer in range 0–7.
- p) Drop_eligible value for Priority Code Point 6: Boolean.
- q) Priority value for Priority Code Point 7: Integer in range 0–7.
- r) Drop_eligible value for Priority Code Point 7: Boolean.

12.16.4.8.3 Outputs

None.

12.16.4.9 Read PIP Priority Code Point Encoding Table

12.16.4.9.1 Purpose

To read the current contents of a row in the Priority Code Point Encoding Table (6.10.3) for a Port.

12.16.4.9.2 Inputs

- a) PIP Index—an identifier for this PIP.
- b) Priority Code Point Row: this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D

12.16.4.9.3 Outputs

- a) Priority Code Point value for priority 0 with drop_eligible False: Integer in range 0–7.
- b) Priority Code Point value for priority 0 with drop_eligible True: Integer in range 0–7.
- c) Priority Code Point value for priority 1 with drop_eligible False: Integer in range 0–7.
- d) Priority Code Point value for priority 1 with drop_eligible True: Integer in range 0–7.
- e) Priority Code Point value for priority 2 with drop_eligible False: Integer in range 0–7.
- f) Priority Code Point value for priority 2 with drop_eligible True: Integer in range 0–7.
- g) Priority Code Point value for priority 3 with drop_eligible False: Integer in range 0–7.
- h) Priority Code Point value for priority 3 with drop_eligible True: Integer in range 0–7.
- i) Priority Code Point value for priority 4 with drop_eligible False: Integer in range 0–7.
- j) Priority Code Point value for priority 4 with drop_eligible True: Integer in range 0–7.
- k) Priority Code Point value for priority 5 with drop_eligible False: Integer in range 0–7.
- l) Priority Code Point value for priority 5 with drop_eligible True: Integer in range 0–7.
- m) Priority Code Point value for priority 6 with drop_eligible False: Integer in range 0–7.
- n) Priority Code Point value for priority 6 with drop_eligible True: Integer in range 0–7.
- o) Priority Code Point value for priority 7 with drop_eligible False: Integer in range 0–7.
- p) Priority Code Point value for priority 7 with drop_eligible True: Integer in range 0–7.

12.16.4.10 Set PIP Priority Code Point Encoding Table (optional)

12.16.4.10.1 Purpose

To modify the contents of a row in the Priority Code Point Encoding Table (6.10.3) for a PIP.

12.16.4.10.2 Inputs

- a) PIP Index—an identifier for this PIP.
- b) Priority Code Point Row—this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D
- c) Priority Code Point value for priority 0 with drop_eligible False: Integer in range 0–7.
- d) Priority Code Point value for priority 0 with drop_eligible True: Integer in range 0–7.

- e) Priority Code Point value for priority 1 with drop_eligible False: Integer in range 0–7.
- f) Priority Code Point value for priority 1 with drop_eligible True: Integer in range 0–7.
- g) Priority Code Point value for priority 2 with drop_eligible False: Integer in range 0–7.
- h) Priority Code Point value for priority 2 with drop_eligible True: Integer in range 0–7.
- i) Priority Code Point value for priority 3 with drop_eligible False: Integer in range 0–7.
- j) Priority Code Point value for priority 3 with drop_eligible True: Integer in range 0–7.
- k) Priority Code Point value for priority 4 with drop_eligible False: Integer in range 0–7.
- l) Priority Code Point value for priority 4 with drop_eligible True: Integer in range 0–7.
- m) Priority Code Point value for priority 5 with drop_eligible False: Integer in range 0–7.
- n) Priority Code Point value for priority 5 with drop_eligible True: Integer in range 0–7.
- o) Priority Code Point value for priority 6 with drop_eligible False: Integer in range 0–7.
- p) Priority Code Point value for priority 6 with drop_eligible True: Integer in range 0–7.
- q) Priority Code Point value for priority 7 with drop_eligible False: Integer in range 0–7.
- r) Priority Code Point value for priority 7 with drop_eligible True: Integer in range 0–7.

12.16.4.10.3 Outputs

None.

12.16.4.11 Read PIP Use_DEI parameter

12.16.4.11.1 Purpose

To read the current state of the Use_DEI parameter (6.10.3) for the PIP.

12.16.4.11.2 Inputs

- a) PIP Index—an identifier for this PIP.

12.16.4.11.3 Outputs

- a) Use_DEI parameter: Boolean.

12.16.4.12 Set PIP Use_DEI parameter (optional)

12.16.4.12.1 Purpose

To set the current state of the Use_DEI parameter (6.10.3) for the PIP.

12.16.4.12.2 Inputs

- a) PIP Index—an identifier for this PIP.
- b) Use_DEI parameter: Boolean.

12.16.4.12.3 Outputs

None.

12.16.4.13 Read PIP Require Drop Encoding parameter

12.16.4.13.1 Purpose

To read the current state of the Require Drop Encoding parameter (8.6.6) for the PIP.

12.16.4.13.2 Inputs

- a) PIP Index—an identifier for this PIP.

12.16.4.13.3 Outputs

- a) Require Drop Encoding parameter—Boolean.

12.16.4.14 Set PIP Require Drop Encoding parameter (optional)**12.16.4.14.1 Purpose**

To set the current state of the Require Drop Encoding parameter (8.6.6) for the PIP.

12.16.4.14.2 Inputs

- a) PIP Index—an identifier for this PIP.
- b) Require Drop Encoding parameter—Boolean.

12.16.4.14.3 Outputs

None.

12.16.5 CBP Configuration managed object

The CBP Configuration managed object includes the following:

- a) A Backbone Service Instance table (6.11)

The management operations that can be performed are as follows:

- c) Read Backbone Service Instance table entry (12.16.5.1)
- d) Set Backbone Service Instance table entry (12.16.5.2)
- e) TESI assignment managed object (12.16.5.3)

12.16.5.1 Read Backbone Service Instance table entry**12.16.5.1.1 Purpose**

All BEBs shall implement the read Backbone Service Instance table entry function to allow reading the Backbone Service Instance table entries in the CBP configuration managed object.

12.16.5.1.2 Inputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component's Port Number for the Bridge Port.
- c) Backbone-SID—identifying the backbone service instance (6.11).

12.16.5.1.3 Outputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component's Port Number for the Bridge Port.
- c) Backbone-SID—the identifier used in the PBBN for the backbone service instance (6.11).
- d) B-VID (optional)—12-bit VID (6.11).

- e) Default Backbone Destination (optional)—EUI-48 DA (6.11).
- f) Ingress, egress, both ingress and egress—a 2-bit selector that determines if frames on this CBP may ingress the PBBN but not egress the PBBN, egress the PBBN but not ingress the PBBN, or both ingress and egress the PBBN. This feature is used to support asymmetric VLANs (F.1.3).
- g) Local-SID (optional)—24-bit I-SID value (6.11).

12.16.5.2 Set Backbone Service Instance table entry

12.16.5.2.1 Purpose

All BEBs shall implement the set Backbone Service Instance table entry function to allow configuration of the backbone service instances available on each CBP. This information is persistent over power-up or reboot.

12.16.5.2.2 Inputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component’s Port Number for the Bridge Port.
- c) Backbone-SID—24-bit I-SID value (6.11).
- d) B-VID (optional)—12-bit VID (6.11).
- e) Default Backbone Destination (optional)—EUI-48 DA (6.11).
- f) Ingress, egress, both ingress and egress—a 2-bit selector that determines if frames on this CBP may ingress to the PBBN but not egress the PBBN, egress to the PBBN but not ingress the PBBN, or both ingress and egress the PBBN. This feature is used to support asymmetric VLANs (F.1.3).
- g) Local-SID (optional)—24-bit I-SID value (6.11).

12.16.5.2.3 Outputs

- a) Operation status—this takes one of the following values:
 - 1) Operation rejected because the ComponentID and Port Number do not identify a valid CBP.
 - 2) Operation accepted.

12.16.5.3 TESI assignment managed object

12.16.5.3.1 Purpose

All CBPs providing TESIs shall implement the TESI assignment managed object to assign a particular TESI on this CBP.

12.16.5.3.2 Inputs

- a) ComponentID—the number identifying the bridge component associated with this port.
- b) Port Number—that component’s Port Number for the Bridge Port.
- c) A series of ordered 3-tuples of the form <ESP-DA, ESP-SA, ESP-VID>.

12.16.5.3.3 Outputs

- a) Operation status—this takes one of the following values:
 - 1) Operation rejected because the ComponentID and Port Number do not identify a valid CBP;
 - 2) Operation rejected because a 3-tuple in the provided series has a VID value in its ESP-VID field that is not allocated to TE-MSTID;
 - 3) Operation rejected because no 3-tuple in the provided series has the CBP’s MAC address in its ESP-SA field;

- 4) Operation rejected because there are only two 3-tuples provided and the values in their ESP-DA and ESP-SA fields are not reversed;
- 5) Operation rejected because there are more than two different values specified in the ESP-DA fields of the provided series of 3-tuples;
- 6) Operation accepted.

12.17 DDCFM entities

The following are the DDCFM managed objects in a Bridge:

- b) DDCFM Stack managed object (12.17.1)
- c) Reflection Responder managed object (12.17.2)
- d) RFM Receiver managed object (12.17.3)
- e) Decapsulator Responder managed object (12.17.4)
- f) SFM Originator managed object (12.17.5)

12.17.1 DDCFM Stack managed object

There is one DDCFM Stack managed object per Bridge. It allows the network administrator to retrieve DDCFM entities configured on a particular Bridge Port or an aggregated IEEE802.3 port within a Bridge Port. The management operation that can be performed is

- a) Read DDCFM Stack managed object (12.17.1.1)

12.17.1.1 Read DDCFM Stack managed object

12.17.1.1.1 Purpose

To retrieve DDCFM entities configured on a particular Bridge Port or an aggregated IEEE802.3 port within a Bridge Port.

12.17.1.1.2 Input

- a) An interface, either a Bridge Port, or an aggregated IEEE802.3 port within a Bridge Port

12.17.1.1.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference to an interface;
 - 2) Operation accepted, but there is no DDCFM entity configured on the specified interface; or
 - 3) Operation accepted, there are (is a) DDCFM entities (entity) configured on the specified interface, and
- b) If there are DDCFM entities configured on the specified interface, the following attributes should be returned:
 - 1) An RR is configured, its associated maintenance domain, and the value indicating its direction; and/or
 - 2) An RFM Receiver is configured, its associated maintenance domain; and/or
 - 3) A DR is configured, its associated maintenance domain, and its associated maintenance association; and/or
 - 4) An SFM Originator is configured, its associated maintenance domain, its associated maintenance association, and the value indicating its direction.

12.17.2 Reflection Responder managed object

The management operations that can be performed on the Reflection Responder managed object are as follows:

- a) Create Reflection Responder managed object (12.17.2.1)
- b) Write Reflection Responder managed object's attributes (12.17.2.2)
- c) Read Reflection Responder managed object's attributes (12.17.2.3)
- d) Delete Reflection Responder managed object (12.17.2.4)
- e) Activate Reflection Responder (12.17.2.5)
- f) Deactivate Reflection Responder (12.17.2.6)

12.17.2.1 Create Reflection Responder managed object

12.17.2.1.1 Purpose

To create a Reflection Responder. After the creation, all the writable attributes have their default values unless being configured differently by the Write Reflection Responder managed object operation (12.17.2.2). The Reflection Responder is in the inactive state after the creation and remains so until activated by a subsequent Activate Reflection Responder operation (12.17.2.5).

12.17.2.1.2 Inputs

- a) A reference to a particular RR managed object, which includes the following:
 - 1) An interface (either a Bridge Port or an aggregated IEEE802.3 port within a Bridge Port) on which the Reflection Responder is defined. The interface identifier could be MAC address or port identifier.
 - 2) A reference to a particular Maintenance Domain managed object (12.14.5).
 - 3) A value indicating the direction in which the RR faces, which can be either Down [Active SAP is further away from the Frame filtering entity (default)], or Up (Active SAP is closer to the Frame filtering entity).

12.17.2.1.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid interface reference or invalid Maintenance Domain level; or
 - 2) Operation accepted.

12.17.2.2 Write Reflection Responder managed object's attributes

12.17.2.2.1 Purpose

To configure and alter the value of one or more writable attribute(s) of an RR managed object. The attributes of an RR managed object cannot be modified when the RR is active.

12.17.2.2.2 Inputs

- a) Reference to a RR managed object [12.17.2.1.2, item a)].
- b) Writable attributes are as follows:
 - 1) Reference to a Maintenance Association managed object (12.14.6) within the RR's Maintenance Domain. If set to 0, the MA associated with the filtered frame shall be used for the RFM emitted by the RR.

- 2) Reflection Filter definition(s), which is to specify what data frames are selected to be reflected. Multiple filters can be combined together by “&& (and)”, “|| (or)”, or “! (negation)” operations with precedence association being determined by “().” DDCFM-capable bridges shall support each of the following filter primitives. Implementers may support additional filters, such as length of data frame being equal, larger or smaller than a particular value.
 - All;
 - VID== vid; This primitive uses SVID if the RR is defined on a S component, and CVID if the RR is defined on a C component. Untagged frame shall not be selected by this filter primitive;
 - DA == xx.xx.xx.xx.xx.xx;
 - SA == xx.xx.xx.xx.xx.xx;
 - The first 2 bytes of the EtherType field ==xx;
- 3) Sampling Interval (29.2.3.4).
- 4) Reflection Target Address, which is a MAC address to which the reflected frames are targeted. Only individual address is allowed for the Reflection Target Address. The default is the source_address of the selected data frame being used for Reflection Target Address.
- 5) Continue option, indicating whether or not the selected data frames are to be continued toward the destination_address specified in the data frame. The default is true to allow data frames to be forwarded to their destinations.
- 6) Boolean variable indicating if the duration is in seconds or in number of data frames.
- 7) Duration in seconds for the Reflection Responder to remain active after being activated. Minimum 1 and maximum is implementation dependent.
- 8) The number data frames for the Refection Responder to reflect after being activated. Once this frame count is reached, the Refection Responder shall be deactivated automatically. Default is 0, which means the duration is limited by RR's timer.
- 9) The priority and drop_eligible parameters to be used in the transmitted encapsulated frames (default value: the highest priority, i.e. that with the highest numerical value, allowed to pass through the Bridge Port for any of the VID. Default value for drop_eligible is false).
- 10) FloodingEnabled Flag indicating if flooding is allowed or not if Egress port cannot be identified for RFM by the Filtering Database.
- 11) Truncation flag indicating if the received data frame should be truncated to keep the length of RFM encapsulated frame not exceeding the Maximum Service Data Unit Size (6.5.8). If the Truncation flag is not set and the RFM encapsulated frame exceeds the Maximum Service Data Unit Size, the filtered data frame is fragmented to two smaller frames and encapsulated in two separate RFMs.

12.17.2.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because the referenced RR managed object does not exist;
 - 2) Operation rejected because of one or more invalid writable attribute(s);
 - 3) Operation rejected because the referenced RR is active; or
 - 4) Operation accepted.

12.17.2.3 Read Reflection Responder managed object's attributes

12.17.2.3.1 Purpose

To retrieve values of attributes in a Reflection Responder managed object.

12.17.2.3.2 Input

- a) Reference to a particular RR managed object [12.17.2.1.2 item a)]

12.17.2.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference to the RR managed object;
 - 2) Operation rejected because there is no Reflection Responder with the given reference; or
 - 3) Operation accepted.
- b) If the operation is accepted, the following attributes for the Responder should be returned:
 - 1) Reference to the RR managed object;
 - 2) Reference to a Maintenance Association managed object (12.14.6) for the RFMs generated by the RR (writable);
 - 3) Reflection Filter definition(s) (writable);
 - 4) Sampling Interval (writable);
 - 5) The Reflection Target Address (writable);
 - 6) Continue option (writable);
 - 7) Duration for Reflection Responder to stay active after being activated (writable);
 - 8) DurationInTimeFlag (writable);
 - 9) The priority and drop_eligible parameters to be used in the transmitted RFMs (writable);
 - 10) FloodingFlag (writable);
 - 11) Truncation flag (writable);
 - 12) Activation status (true or false);
 - 13) Remaining time or count left for the RR to be active; and
 - 14) nextRFMtransID (29.3.1.16) value.

12.17.2.4 Delete Reflection Responder managed object

12.17.2.4.1 Purpose

To delete a Reflection Responder managed object. If the corresponding Reflection Responder is still active, the deletion will deactivate the Reflection Responder and then delete the corresponding managed object.

12.17.2.4.2 Input

- a) Reference to a RR managed object [12.17.2.1.2, item a)]

12.17.2.4.3 Outputs

- a) Operation status: this takes one of the following values:
 - 1) Operation rejected because of invalid reference to RR;
 - 2) Operation rejected because the referenced RR managed does not exist; or
 - 3) Operation accepted.

12.17.2.5 Activate Reflection Responder

12.17.2.5.1 Purpose

To activate a Reflection Responder.

Depending on the filter conditions specified for each RR, some RRs can potentially reflect large amount of data frames into the network, which could cause excessive extra traffic injected into the network and impact network performance. For those RRs, it is recommended that no more than one instance of the Reflection Responder (RR) managed object be activated per MD for an RR associated with an MP and that no more than one instance of the RR managed object be activated per VID for an RR not associated with an MP (e.g., for an RR created on a bridge port).

12.17.2.5.2 Inputs

- a) Reference to a RR managed object [12.17.2.1.2 a)].

12.17.2.5.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because invalid reference or the referenced RR does not exist;
 - 2) Operation rejected because of the referenced RR already being activated; or
 - 3) Operation accepted.

12.17.2.6 De-Activate Reflection Responder

12.17.2.6.1 Purpose

To deactivate an active Reflection Responder before its duration expires.

12.17.2.6.2 Input

- a) Reference to a RR managed object [12.17.2.1.2, item a)].

12.17.2.6.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because invalid reference or the referenced RR does not exist;
 - 2) Operation rejected because of the referenced RR not being activated; or
 - 3) Operation accepted.

12.17.3 RFM Receiver managed object

The management operations that can be performed on the RFM Receiver managed object are as follows:

- a) Create RFM Receiver managed object (12.17.3.1)
- b) Delete RFM Receiver managed object (12.17.3.2)

An RFM Receiver is active when the corresponding RFM Receiver managed object is created. Therefore, there is no “Activate and De-activate” operations for the RFM Receiver managed object.

12.17.3.1 Create RFM Receiver managed object

12.17.3.1.1 Purpose

To create an RFM Receiver managed object. Once an RFM Receiver managed object is created, the corresponding RFM Receiver is able to receive RFMs.

12.17.3.1.2 Inputs

- a) Reference to an RFM Receiver managed object, which could be either
 - 1) Reference to an MP; or
 - 2) An interface (either a Bridge Port or an aggregated IEEE802.3 port within a Bridge Port) on which RFM Receiver is created and a reference to the corresponding RR’s Maintenance Domain managed object.

12.17.3.1.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference to the RFM Receiver managed object;
 - 2) Operation rejected because the referenced RFM Receiver managed object already exists; or
 - 3) Operation accepted.

12.17.3.2 Delete RFM Receiver managed object

12.17.3.2.1 Purpose

To delete an RFM Receiver managed object.

12.17.3.2.2 Input

- a) Reference to an RFM Receiver managed object [12.17.3.1.2, item a)]

12.17.3.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference to the RFM Receiver managed object;
 - 2) Operation rejected because the referenced RFM Receiver managed object does not exist; or
 - 3) Operation accepted.

12.17.4 Decapsulator Responder managed object

The management operations that can be performed on the Decapsulator Responder managed object are as follows:

- a) Create Decapsulator Responder managed object (12.17.4.1)
- b) Read Decapsulator Responder managed object (12.17.4.2)
- c) Write Decapsulator Responder managed object's attributes (12.17.4.3)
- d) Delete Decapsulator Responder managed object (12.17.4.4)
- e) Activate Decapsulator Responder (12.17.4.5)
- f) De-activate Decapsulator Responder (12.17.4.6)

12.17.4.1 Create Decapsulator Responder managed object

12.17.4.1.1 Purpose

To create a Decapsulator Responder managed object and corresponding Decapsulator Responder.

12.17.4.1.2 Inputs

- a) Reference to a Decapsulator Responder, which could be either
 - 1) Reference to an MP; or
 - 2) A triple consisting of an interface (either a Bridge Port or an aggregated IEEE802.3 port within a Bridge Port), a reference to a Maintenance Domain managed object, and a reference to a Maintenance Association managed object.

12.17.4.1.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference to a Decapsulator Responder managed object;

- 2) Operation rejected because the referenced Decapsulator Responder managed object already exists; or
- 3) Operation accepted.

12.17.4.2 Read Decapsulator Responder managed object

12.17.4.2.1 Purpose

To retrieve the attributes of a Decapsulator Responder managed object.

12.17.4.2.2 Input

- a) Reference to a Decapsulator Responder managed object [12.17.4.1.2, item a)].

12.17.4.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference to a Decapsulator Responder managed object;
 - 2) Operation rejected because the referenced Decapsulator Responder managed object does not exist; or
 - 3) Operation accepted.
- b) If the operation is accepted, the following attributes for the Decapsulator Responder should be returned:
 - 1) Reference to the Decapsulator Responder managed object;
 - 2) MAC address of the SFM Originator (writable);
 - 3) The value of SourceAddressStayFlag (writable);
 - 4) The value of FloodingEnabled flag (writable);
 - 5) The specified duration in seconds, for Decapsulator Responder to remain active once activated (writable);
 - 6) Activation status;
 - 7) Remaining time left for Decapsulator Responder to be active; and
 - 8) The total number of out-of-sequence SFMs received (29.3.8.7).

12.17.4.3 Write Decapsulator Responder managed object's attributes

12.17.4.3.1 Purpose

To alter the value of one or more attributes of a Decapsulator Responder managed object when the corresponding Decapsulator Responder is not active.

12.17.4.3.2 Inputs

- a) Reference to a Decapsulator Responder managed object [12.17.4.1.2, item a)];
- b) Reference to the attribute whose value is to be changed;
 - 1) Boolean flag, SourceAddressStayFlag, to enforce the DR not to replace the source_address field of the decapsulated frame with the DR's own MAC address. Default is false;
 - 2) MAC address of the SFM Originator, which is optional;
 - 3) Boolean FloodingEnabled flag indicating if flooding is allowed (true) or not (false) if the Egress port cannot be identified by the Filtering Database; and
 - 4) Duration in seconds, for Decapsulator Responder to remain active once activated.

12.17.4.3.3 Outputs

- a) Operation status. This takes one of the following values:

- 1) Operation rejected because of invalid reference to a Decapsulator Responder managed object;
- 2) Operation rejected because the referenced Decapsulator Responder managed object does not exist;
- 3) Operation rejected because of invalid attribute reference or invalid value;
- 4) Operation rejected because the referenced Decapsulator Responder is active; or
- 5) Operation accepted.

12.17.4.4 Delete Decapsulator Responder managed object

12.17.4.4.1 Purpose

To delete a Decapsulator Responder managed object. If the Decapsulator Responder is still active, the deletion will deactivate the Decapsulator Responder first and then delete the corresponding managed object.

12.17.4.4.2 Input

- a) Reference to a Decapsulator Responder managed object [12.17.4.1.2, item a)].

12.17.4.4.3 Outputs

- a) Operation status: this takes one of the following values:
 - 1) Operation rejected because of invalid reference to a Decapsulator Responder managed object;
 - 2) Operation rejected because the referenced Decapsulator Responder managed object does not exist; or
 - 3) Operation accepted.

12.17.4.5 Activate a Decapsulator Responder

12.17.4.5.1 Purpose

To activate a Decapsulator Responder.

12.17.4.5.2 Input

- a) Reference to the Decapsulator Responder managed object [12.17.4.1.2, item a)].

12.17.4.5.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced Decapsulator Responder managed object does not exist;
 - 3) Operation rejected because the referenced Decapsulator Responder has already been activated; or
 - 4) Operation accepted.

12.17.4.6 De-activate a Decapsulator Responder

12.17.4.6.1 Purpose

To deactivate a Decapsulator Responder.

12.17.4.6.2 Input

- a) Reference to the Decapsulator Responder managed object [12.17.4.1.2, item a)].

12.17.4.6.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced Decapsulator Responder managed object does not exist;
 - 3) Operation rejected because the referenced Decapsulator Responder is not active; or
 - 4) Operation accepted.

12.17.5 SFM Originator managed object

The management operations that can be performed on the SFM Originator managed object are as follows:

- a) Create SFM Originator managed object (12.17.5.1)
- b) Read SFM Originator managed object (12.17.5.2)
- c) Delete SFM Originator managed object (12.17.5.3)
- d) Write SFM Originator managed object's attributes (12.17.5.4)
- e) Activate SFM Originator (12.17.5.5)
- f) De-activate SFM Originator (12.17.5.6)

12.17.5.1 Create SFM Originator managed object

12.17.5.1.1 Purpose

To create a SFM Originator managed object.

12.17.5.1.2 Inputs

- a) Reference to a SFM Originator managed object, which could be either
 - 1) Reference to an MP; or
 - 2) An interface (either a Bridge Port, or an aggregated IEEE802.3 port within a Bridge Port), a reference to a Maintenance Domain managed object, a reference to a Maintenance Association managed object, and a value indicating the direction in which the SFM Originator faces on the interface.

12.17.5.1.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced SFM Originator managed object already exists; or
 - 3) Operation accepted.

12.17.5.2 Read SFM Originator managed object

12.17.5.2.1 Purpose

To retrieve attributes of SFM Originator managed object.

12.17.5.2.2 Inputs

- a) Reference to a SFM Originator managed object [12.17.5.1.2, item a)].

12.17.5.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced SFM Originator managed object does not exist; or
 - 3) Operation accepted.
- b) If the operation is accepted, following attributes for the SFM Originator should be returned:
 - 1) MAC address of the SFM Originator;
 - 2) MAC address of the corresponding Decapsulator Responder (writable);
 - 3) Duration (writable);
 - 4) Activation status: true/false; and
 - 5) The remaining time left for the SFM Originator to remain active.

12.17.5.3 Delete SFM Originator managed object

12.17.5.3.1 Purpose

To delete a SFM Originator managed object, and the corresponding SFM Originator.

12.17.5.3.2 Inputs

- a) The reference to a SFM Originator managed object [12.17.5.1.2, item a)].

12.17.5.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced SFM Originator managed object does not exist; or
 - 3) Operation accepted.

12.17.5.4 Write SFM Originator managed object's attribute

12.17.5.4.1 Purpose

To change an attribute of a SFM Originator managed object.

12.17.5.4.2 Inputs

- a) Reference to a SFM Originator managed object [12.17.5.1.2, item a)];
- b) MAC address of the corresponding Decapsulator Responder; and
- c) Duration in seconds, for SFM Originator to stay active once activated.

12.17.5.4.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference of SFM Originator managed object or the attributes;
 - 2) Operation rejected because the referenced SFM Originator managed object does not exist; or
 - 3) Operation accepted.

12.17.5.5 Activate SFM Originator

12.17.5.5.1 Purpose

To activate a SFM Originator.

12.17.5.5.2 Input

- a) Reference to the SFM Originator managed object [12.17.5.1.2, item a)].

12.17.5.5.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced SFM Originator managed object does not exist;
 - 3) Operation rejected because the referenced SFM Originator has already been activated; or
 - 4) Operation accepted.

12.17.5.6 De-activate SFM Originator

12.17.5.6.1 Purpose

To deactivate a SFM Originator.

12.17.5.6.2 Input

- a) Reference to a SFM Originator managed object [12.17.5.1.2, item a)].

12.17.5.6.3 Output

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected because of invalid reference;
 - 2) Operation rejected because the referenced SFM Originator managed object does not exist;
 - 3) Operation rejected because the referenced SFM Originator is not active; or
 - 4) Operation accepted.

12.18 PBB-TE Protection Switching managed objects

The PBB-TE Protection Switching managed objects model operations that create, modify, delete, or inquire about, the configuration and the operation of protection switching. The following are the PBB-TE Protection Switching managed objects in a B-component of an IB-BEB capable of supporting TESIs:

- a) The TE protection group list managed object (12.18.1)
- b) The TE protection group managed object (12.18.2)

12.18.1 TE protection group list managed object

There is one TE protection group list managed object per IB-BEB. It contains a list of the TE protection groups that have been configured on the IB-BEB.

The management operations that can be performed on the TE protection group list managed object are as follows:

- a) Read TE protection group list (12.18.1.1)

- b) Create TE protection group managed object (12.18.1.2)
- c) Delete TE protection group managed object (12.18.1.3)

12.18.1.1 Read TE protection group list

12.18.1.1.1 Purpose

To obtain information about all of the TE protection groups in an IB-BEB.

12.18.1.1.2 Inputs

None.

12.18.1.1.3 Outputs

A list, perhaps empty, of the TE protection group managed objects configured on the IB-BEB. For each item in the list, the Read TE protection group list command returns:

- a) A reference to a particular Maintenance Association managed object (12.14.6) identifying a PBB-TE MA that corresponds to the group's working entity; and
- b) A reference to a particular Maintenance Association managed object (12.14.6) identifying a PBB-TE MA with the same end points as the one provided in item b) that corresponds to the group's protection entity.
- c) A reference to all TE protection group managed objects (12.18.2) that share a working or protection entity with the TE protection group in the list.

12.18.1.2 Create TE protection group managed object

12.18.1.2.1 Purpose

To create a new TE protection group managed object in an IB-BEB and add it to the IB-BEB's TE protection group list managed object.

12.18.1.2.2 Inputs

- a) A reference to a particular Maintenance Association managed object (12.14.6) identifying a PBB-TE MA that corresponds to the working entity; and
- b) A reference to a particular Maintenance Association managed object (12.14.6) identifying a PBB-TE MA with the same end points as the one provided in item a) that will monitor the protection entity.

12.18.1.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to nonexistent Maintenance Domain;
 - 2) Operation rejected due to nonexistent Maintenance Associations;
 - 3) Operation rejected because the referred MAs do not correspond to PBB-TE services;
 - 4) Operation rejected because the two referred MAs do not have the same end points;
 - 5) Operation rejected because the two referred MAs do not belong to the same MD;
 - 6) Operation rejected due to duplicate MAs; or
 - 7) Operation accepted. In addition, the list, perhaps empty, of the TE protection group managed objects (12.18.2) that share a working or protection entity with the created TE protection group is provided.

12.18.1.3 Delete TE protection group managed object

12.18.1.3.1 Purpose

To delete a specific TE protection group managed object from an IB-BEB and from the IB-BEB's TE protection group list managed object.

12.18.1.3.2 Inputs

- a) A reference to a particular TE protection group managed object (12.18.2).

12.18.1.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to nonexistent TE protection group;
 - 2) Operation rejected because the TE protection group is enabled; or
 - 3) Operation accepted.

12.18.2 TE protection group managed object

There can be any number of TE protection group managed objects per IB-BEB, one for each working and protection pair of TESIs with common end points.

The management operations that can be performed on the TE protection group managed object are as follows:

- a) Read TE protection group managed object (12.18.2.1);
- b) Write TE protection group managed object (12.18.2.2); and
- c) TE protection group administrative commands managed object (12.18.2.3).

12.18.2.1 Read TE protection group managed object

12.18.2.1.1 Purpose

To obtain information from an IB-BEB about a specific TE protection group managed object.

12.18.2.1.2 Inputs

- a) A reference to a particular TE protection group managed object (12.18.2).

12.18.2.1.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to nonexistent TE protection group; or
 - 2) Operation accepted;
- b) (writable) A list of backbone-SIDs, identifying the protected backbone service instances, or NULL indicating that no backbone service instances are assigned to the TE protection group and that the TE protection group is disabled.
- c) An enumerated value indicating the operational state of the Service Mapping state machine (Figure 26-15) for each TE protection group:
 - 1) WORKING_PATH;
 - 2) PROTECTION_PATH;
 - 3) WTR; or
 - 4) PROT_ADMIN.

- d) An enumerated value indicating the status of active requests within the TE protection group:
 - 1) **NoRequest.** No administrative command is in effect;
 - 2) **LoP.** Set if LoP (26.10.3.3.4) is TRUE, indicating that an administrative command to prohibit the use of the protection entity is in effect;
 - 3) **FS.** Set if FS (26.10.3.3.5) is TRUE, indicating that an administrative command to perform forced switching to the protection path(s) is in effect;
 - 4) **p.SFH.** Set if SFH (26.10.3.3.3) is TRUE on the protection entity;
 - 5) **w.SFH.** Set if SFH (26.10.3.3.3) is TRUE on the working entity;
 - 6) **MStoProtection.** Set if MStoProtection (26.10.3.3.6) is TRUE, indicating that an administrative command to perform manual switching to the protection path(s) is in effect;
 - 7) **MStoWorking.** Set if MStoWorking (26.10.3.3.7) is TRUE, indicating that an administrative command to perform manual switching to the working path is in effect;
- e) (writable) (optional) The Wait-to-restore (WTR) period (26.10.3.3.8). In revertive operation it may be configured in steps of 1 min between 5 min and 12 min; the default value is 5 min. The value 0 indicates nonrevertive operation; and
- f) (writable) (optional) The Hold-Off period (26.10.3.3.9). The range of the Hold-Off period is 0 s to 10 s in steps of 100 ms; the default value is 0.

12.18.2.2 Write TE protection group managed object

12.18.2.2.1 Purpose

To alter a value of a specific TE protection group managed object.

12.18.2.2.2 Inputs

- a) A reference to a particular TE protection group managed object (12.18.2);
- b) A reference to one of the writable managed objects in 12.18.2.1.3 that is to be altered; and
- c) A new value for the managed object.

12.18.2.2.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to nonexistent TE protection group;
 - 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;
 - 5) Operation accepted.

12.18.2.3 TE protection group administrative commands managed object

12.18.2.3.1 Purpose

To set an administrative command on a TE protection group managed object.

12.18.2.3.2 Inputs

- a) A reference to a particular TE protection group managed object (12.18.2); and
- b) An enumerated value indicating the exercised administrative command:
 - 1) **Clear.** An indication to clear all other administrative commands;
 - 2) **Lockout of Protection.** An administrative command to prohibit the use of the protection entity;
 - 3) **Forced Switch.** An administrative command to perform forced switching to the protection path(s);

- 4) **Manual Switch To Protection.** An administrative command to perform manual switching to the protection path(s) if it is operational;
- 5) **Manual Switch To Working.** An administrative command to perform manual switching to the working path if it is operational.

12.18.2.3.3 Outputs

- a) Operation status. This takes one of the following values:
 - 1) Operation rejected due to nonexistent TE protection group;
 - 2) Operation rejected due to lack of authority to set this variable;
 - 3) Operation rejected due to an active higher priority request; or
 - 4) Operation accepted.

12.19 TPMR managed objects

The managed resources of a TPMR, accessed via the TPMR management entity, are those of the Processes and Entities defined in this clause, and listed as follows:

- a) The TPMR management entity (12.19.1)
- b) The individual MAC and PHY entities associated with each TPMR Port
- c) The Forwarding Process of the MAC Relay Entity (8.6, 12.19.3)
- d) The MAC status propagation entity (23.3, 12.19.4)

The management of each of these resources is described in terms of managed objects and operations in the following subclauses.

12.19.1 TPMR management entity

The objects that comprise this managed resource are as follows:

- a) The TPMR Configuration (12.19.1.1)
- b) The Port Configuration for each Port (12.19.1.2)

12.19.1.1 TPMR configuration

The TPMR configuration object models the operations that modify, or inquire about, the configuration of the TPMR's resources. There is a single TPMR configuration object per TPMR.

The management operations that can be performed on the TPMR configuration are as follows:

- a) Read TPMR (12.19.1.1.1)
- b) Set TPMR Name (12.19.1.1.2)

12.19.1.1.1 Read TPMR

12.19.1.1.1.1 Purpose

To obtain general information regarding the TPMR.

12.19.1.1.1.2 Inputs

None.

12.19.1.1.3 Outputs

- a) TPMR Name—a text string of up to 32 characters, of locally determined significance.
- b) TPMR MAC addresses—the MAC addresses of the two externally accessible TPMR Ports. The following information is returned for each Port:
 - 1) The Port number.
 - 2) The MAC address of the Port.
 - 3) A Boolean value, which is TRUE if the MAC address is the management address for the TPMR, and is otherwise FALSE.

NOTE 1—The TPMR management entity may make use of one or both Ports of a TPMR to transmit and receive management frames. However, the MAC address used by the TPMR management entity as the source MAC address in transmitted management frames (the management MAC address) is the individual MAC address associated with one of the Ports of the TPMR (a management Port, see 8.3, 8.5).

NOTE 2—As the transmission and reception rules for a TPMR (8.5.2) mean that frames forwarded by a TPMR are not received by higher layer entities on the forwarding Port, by receiving management frames on either Port the TPMR management entity can determine which LAN incoming management frames were transmitted on, as incoming frames are only presented to higher layer entities attached to the reception Port.

- c) Uptime—count in seconds of the time elapsed since the TPMR was last reset or initialized.

12.19.1.1.2 Set TPMR Name

12.19.1.1.2.1 Purpose

To associate a text string, readable by the Read TPMR operation, with a TPMR.

12.19.1.1.2.2 Inputs

- a) TPMR Name—a text string of up to 32 characters.

12.19.1.1.2.3 Outputs

None.

12.19.1.2 Port configuration

The Port Configuration object models the operations that modify, or inquire about, the configuration of the Ports of a TPMR. By definition there are two Ports per TPMR (one for each MAC interface), and each is identified by a permanently allocated Port number.

The management operations that can be performed on the Port Configuration are as follows:

- a) Read Port (12.19.1.2.1)
- b) Set Port Name (12.19.1.2.2)

12.19.1.2.1 Read Port

12.19.1.2.1.1 Purpose

To obtain general information regarding a specific TPMR Port.

12.19.1.2.1.2 Inputs

- a) Port Number—the number of the Port. This parameter can take the integer values 1 and 2 only.

12.19.1.2.1.3 Outputs

- a) Port Name—a text string of up to 32 characters, of locally determined significance.
- b) Port Type—the MAC Entity type of the Port (IEEE Std 802.3; ISO/IEC 8802-11; ISO 9314; other).
- c) Management address forwarding status. This takes the value TRUE if forwarding is enabled for the management address on the Port or FALSE if forwarding is disabled for the management address on the Port (see 5.13).

12.19.1.2.2 Set Port Name

12.19.1.2.2.1 Purpose

To associate a text string, readable by the Read Port operation, with a TPMR Port.

12.19.1.2.2.2 Inputs

- a) Port Number—the number of the Port. This parameter can take the integer values 1 and 2 only.
- b) Port Name—a text string of up to 32 characters.

12.19.1.2.2.3 Outputs

None.

12.19.2 MAC and PHY entities

The management operations and facilities provided by the MAC and PHY entities are those specified in the layer management standards of the individual MACs and PHYs. A MAC entity, and its underlying PHY entity, is associated with each TPMR Port.

12.19.3 Forwarding Process

The Forwarding Process contains information relating to the forwarding of frames. Counters are maintained that provide information on the number of frames forwarded, filtered, and dropped due to error.

The objects that comprise this managed resource are as follows:

- a) The Port Counters (12.19.3.1).
- b) Optionally, the Priority Handling objects for each Port (12.19.3.2).
- c) Optionally, The Traffic Class Table for each Port (12.6.3).

12.19.3.1 The Port Counters

The Port Counters object models the operations that can be performed on the Port counters of the Forwarding Process resource. There is one instance of the Port Counters object for each MAC Entity per TPMR (i.e., two instances total).

The management operation that can be performed on the Port Counters is as follows:

- a) Read Forwarding Port Counters (12.19.3.1.1).

12.19.3.1.1 Read forwarding port counters

12.19.3.1.1.1 Purpose

To read the forwarding counters associated with a specific TPMR Port.

12.19.3.1.1.2 Inputs

- a) Port Number—the number of the Port. This parameter can take the integer values 1 and 2 only.

12.19.3.1.1.3 Outputs

- a) Frames Received—count of all valid frames received on this Port (including BPDUs, frames addressed to the TPMR as an end station, and frames that were submitted to the Forwarding Process, 8.6).
- b) Octets Received—count of the total number of octets in all valid frames received on this Port (including BPDUs, frames addressed to the TPMR as an end station, and frames that were submitted to the Forwarding Process).
- c) Frames Discarded by Forwarding Process—count of all frames that were received on this Port but were discarded by the Forwarding Process for any reason.
- d) Frames Forwarded to Transmission Port—count of all frames that were received on this Port and were forwarded to the transmission Port (8.6.8).
- e) Frames Discarded due to Queue Full—count of all frames received on this Port that were to be transmitted through the transmission Port but were discarded due to lack of available queue space (8.6.6, 8.6.7).
- f) Frames Discarded Lifetime Exceeded—count of all frames received on this Port that were to be transmitted through the transmission Port but were discarded due to their frame lifetime having been exceeded (8.6.7).
- g) Frames Discarded on Error—count of all frames received on this Port that were to be transmitted through the transmission Port but could not be transmitted (e.g., frame would be too large, 6.5.8).
- h) Frame Discard on Error Details—a list of 16 elements, each containing the source address of a frame and the reason why the frame was discarded (e.g., frame too large). The list is maintained as a circular buffer. The reason for discard on error, at present, is as follows:
 - 1) Transmissible service data unit size exceeded.

12.19.3.2 Priority handling

The Priority Handling object models the operations that can be performed on, or inquire about, the Default Priority parameter, the Priority Regeneration Table parameter, the Outbound Access Priority Table parameter, the Priority Code Point parameters, and the Service Access Priority parameters for each Port. The operations that can be performed on this object are as follows:

- a) Read Port Default Priority (12.6.2.1).
- b) Set Port Default Priority (12.6.2.2).
- c) Read Port Priority Regeneration Table (12.6.2.3).
- d) Set Port Priority Regeneration Table (12.6.2.4).
- e) Read Port Priority Code Point Selection (12.6.2.5).
- f) Set Port Priority Code Point Selection (12.6.2.6).
- g) Optionally, Read Port Priority Code Point Decoding Table (12.6.2.7).
- h) Read Use_DEI parameter (12.6.2.12).
- i) Optionally, Set Use_DEI parameter (12.6.2.13).

12.19.3.2.1 Read Port Default Priority

12.19.3.2.1.1 Purpose

To read the current state of the Default Priority parameter (6.6) for a specific TPMR Port.

12.19.3.2.1.2 Inputs

- a) Port number.

12.19.3.2.1.3 Outputs

- a) Default Priority value—Integer in range 0–7.

12.19.3.2.2 Set Port Default Priority

12.19.3.2.2.1 Purpose

To set the current state of the Default Priority parameter (6.6) for a specific TPMR Port.

12.19.3.2.2.2 Inputs

- a) Port number.
- b) Default Priority value—Integer in range 0–7.

12.19.3.2.2.3 Outputs

None.

12.19.3.2.3 Read Port Priority Regeneration Table

12.19.3.2.3.1 Purpose

To read the current state of the Priority Regeneration Table parameter (6.9.4) for a specific TPMR Port.

12.19.3.2.3.2 Inputs

- a) Port number.

12.19.3.2.3.3 Outputs

- a) Regenerated Priority value for Received Priority 0—Integer in range 0–7.
- b) Regenerated Priority value for Received Priority 1—Integer in range 0–7.
- c) Regenerated Priority value for Received Priority 2—Integer in range 0–7.
- d) Regenerated Priority value for Received Priority 3—Integer in range 0–7.
- e) Regenerated Priority value for Received Priority 4—Integer in range 0–7.
- f) Regenerated Priority value for Received Priority 5—Integer in range 0–7.
- g) Regenerated Priority value for Received Priority 6—Integer in range 0–7.
- h) Regenerated Priority value for Received Priority 7—Integer in range 0–7.

12.19.3.2.4 Set Port Priority Regeneration Table

12.19.3.2.4.1 Purpose

To set the current state of the Priority Regeneration Table parameter (6.9.4) for a specific TPMR Port.

12.19.3.2.4.2 Inputs

- a) Port number;
- b) Regenerated Priority value for Received Priority 0—Integer in range 0–7.
- c) Regenerated Priority value for Received Priority 1—Integer in range 0–7.
- d) Regenerated Priority value for Received Priority 2—Integer in range 0–7.
- e) Regenerated Priority value for Received Priority 3—Integer in range 0–7.
- f) Regenerated Priority value for Received Priority 4—Integer in range 0–7.
- g) Regenerated Priority value for Received Priority 5—Integer in range 0–7.
- h) Regenerated Priority value for Received Priority 6—Integer in range 0–7.
- i) Regenerated Priority value for Received Priority 7—Integer in range 0–7.

12.19.3.2.4.3 Outputs

None.

12.19.3.2.5 Read Port Priority Code Point Selection

12.19.3.2.5.1 Purpose

To read which row of the Priority Code Point Decoding Table (6.9.3) is currently selected for use on this Port.

12.19.3.2.5.2 Inputs

- a) Port Number: the number of the TPMR Port.

12.19.3.2.5.3 Outputs

- a) Priority Code Point Selection: this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D

12.19.3.2.6 Set Port Priority Code Point Selection

12.19.3.2.6.1 Purpose

To set which row of the Priority Code Point Decoding Table (6.9.3) will be selected for use on this Port.

12.19.3.2.6.2 Inputs

- a) Port Number: the number of the TPMR Port.
- b) Priority Code Point Selection: this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D

- 4) 5P3D

12.19.3.2.6.3 Outputs

None.

12.19.3.2.7 Read Priority Code Point Decoding Table

12.19.3.2.7.1 Purpose

To read the current contents of a row in the Priority Code Point Decoding Table (6.9.3) for a Port.

12.19.3.2.7.2 Inputs

- a) Port Number: the number of the Bridge Port.
- b) Priority Code Point Row: this takes one of the following values:
 - 1) 8P0D
 - 2) 7P1D
 - 3) 6P2D
 - 4) 5P3D

12.19.3.2.7.3 Outputs

- a) Priority value for Priority Code Point 0: Integer in range 0–7.
- b) Drop_eligible value for Priority Code Point 0: Boolean.
- c) Priority value for Priority Code Point 1: Integer in range 0–7.
- d) Drop_eligible value for Priority Code Point 1: Boolean.
- e) Priority value for Priority Code Point 2: Integer in range 0–7.
- f) Drop_eligible value for Priority Code Point 2: Boolean.
- g) Priority value for Priority Code Point 3: Integer in range 0–7.
- h) Drop_eligible value for Priority Code Point 3: Boolean.
- i) Priority value for Priority Code Point 4: Integer in range 0–7.
- j) Drop_eligible value for Priority Code Point 4: Boolean.
- k) Priority value for Priority Code Point 5: Integer in range 0–7.
- l) Drop_eligible value for Priority Code Point 5: Boolean.
- m) Priority value for Priority Code Point 6: Integer in range 0–7.
- n) Drop_eligible value for Priority Code Point 6: Boolean.
- o) Priority value for Priority Code Point 7: Integer in range 0–7.
- p) Drop_eligible value for Priority Code Point 7: Boolean.

12.19.3.2.8 Read Use_DEI Parameter

12.19.3.2.8.1 Purpose

To read the current state of the Use_DEI parameter (6.9.3) for the Port.

12.19.3.2.8.2 Inputs

- a) Port Number: the number of the TPMR Port.

12.19.3.2.8.3 Outputs

- a) Use_DEI parameter: Boolean.

12.19.3.2.9 Set Use_DEI Parameter

12.19.3.2.9.1 Purpose

To set the current state of the Use_DEI parameter (6.9.3) for the Port.

12.19.3.2.9.2 Inputs

- a) Port Number: the number of the TPMR Port.
- a) Use_DEI parameter: Boolean.

12.19.3.2.9.3 Outputs

None.

12.19.3.3 Traffic Class Table

The Traffic Class Table object models the operations that can be performed on, or inquire about, the current contents of the Traffic Class Table (8.6.6) for a given Port. The operations that can be performed on this object are Read Port Traffic Class Table and Set Port Traffic Class Table.

12.19.3.3.1 Read Port Traffic Class Table

12.19.3.3.1.1 Purpose

To read the contents of the Traffic Class Table (8.6.6) for a given Port.

12.19.3.3.1.2 Inputs

- a) Port Number.

12.19.3.3.1.3 Outputs

- a) The number of traffic classes, in the range 1 through 8, supported on the Port.
- b) For each value of traffic class supported on the Port, the value of the traffic class in the range 0 through 7, and the set of priority values assigned to that traffic class.

12.19.3.3.2 Set Port Traffic Class Table

12.19.3.3.2.1 Purpose

To set the contents of the Traffic Class Table (8.6.6) for a given Port.

12.19.3.3.2.2 Inputs

- a) Port number;
- b) For each value of traffic class supported on the Port, the value of the traffic class in the range 0 through 7, and the set of priority values assigned to that traffic class.

NOTE—If a traffic class value greater than the largest traffic class available on the Port is specified, then the value applied to the traffic class Table is the largest available traffic class.

12.19.3.3.2.3 Outputs

None.

12.19.4 MAC status propagation entity

The object that comprises this managed resource is as follows:

- a) The MAC status propagation configuration for each Port of the TPMR (12.19.4.1).

12.19.4.1 MAC status propagation (MSP)

The MAC status propagation object models the operations that modify, or inquire about, the configuration and statistics of the MAC status propagation entity for a particular Port of the TPMR.

The management operations that can be performed on the MAC status propagation configuration are as follows:

- a) Read MSP performance parameters (12.19.4.1.1).
- b) Set MSP performance parameters (12.19.4.1.2).
- c) Read MSP statistics (12.19.4.1.3).

12.19.4.1.1 Read MSP performance parameters

12.19.4.1.1.1 Purpose

To solicit information regarding the current configuration of the MSP performance parameters with respect to a particular Port of the TPMR.

12.19.4.1.1.2 Inputs

- a) Port number. The number of the Port of the TPMR. This can take the values 1 or 2 only.

12.19.4.1.1.3 Outputs

- a) LinkNotify. The current value (Boolean) of LinkNotify (23.5.1) being used by the MSP state machines.
- b) LinkNotifyWait. The current value, in centiseconds, of LinkNotifyWait (23.5.2) being used by the MSP state machines.
- c) LinkNotifyRetry. The current value, in centiseconds, of LinkNotifyRetry (23.5.3) being used by the MSP state machines.
- d) MACNotify. The current value (Boolean) of MACNotify (23.5.4) being used by the MSP state machines.
- e) MACNotifyTime. The current value, in centiseconds, of MACNotifyTime (23.5.5) being used by the MSP state machines.
- f) MACRecoverTime. The current value, in centiseconds, of MACRecoverTime (23.5.6) being used by the MSP state machines.

12.19.4.1.2 Set MSP performance parameters

12.19.4.1.2.1 Purpose

To change the MSP performance parameters to be used for a particular Port of the TPMR.

12.19.4.1.2.2 Inputs

- a) Port number. The number of the Port of the TPMR. This can take the values 1 or 2 only.
- b) LinkNotify. The desired value (Boolean) of LinkNotify (23.5.1) to be used by the MSP state machines.
- c) LinkNotifyWait. The desired value, in centiseconds, of LinkNotifyWait (23.5.2) to be used by the MSP state machines.
- d) LinkNotifyRetry. The desired value, in centiseconds, of LinkNotifyRetry (23.5.3) to be used by the MSP state machines.
- e) MACNotify. The desired value (Boolean) of MACNotify (23.5.4) to be used by the MSP state machines.
- f) MACNotifyTime. The desired value, in centiseconds, of MACNotifyTime (23.5.5) to be used by the MSP state machines.
- g) MACRecoverTime. The desired value, in centiseconds, of MACRecoverTime (23.5.6) to be used by the MSP state machines.

12.19.4.1.2.3 Outputs

None.

12.19.4.1.3 Read MSP statistics

12.19.4.1.3.1 Purpose

To read the MSP statistics (23.12) for a particular Port of the TPMR.

12.19.4.1.3.2 Inputs

- a) Port number. The number of the Port of the TPMR. This can take the values 1 or 2 only.

12.19.4.1.3.3 Outputs

- a) acksTransmitted: The number of *acks* transmitted (23.6.15) by the Port's Transmit Process as a consequence of txAck being set.
- b) addNotificationsTransmitted: The number of *adds* transmitted (23.6.16) by the Port's Transmit Process as a consequence of txAdd being set.
- c) addConfirmationsTransmitted: The number of *add confirms* transmitted (23.6.17) by the Port's Transmit Process as a consequence of txAddConfirm being set.
- d) lossNotificationsTransmitted: The number of *losses* transmitted (23.6.18) by the Port's Transmit Process as a consequence of txLoss being set.
- e) lossConfirmationsTransmitted: The number of *loss confirms* transmitted (23.6.19) by the Port's Transmit Process as a consequence of txLossConfirm being set.
- f) acksReceived: The number of *acks* received (23.6.10) by the Port's Transmit Process.
- g) addNotificationsReceived: The number of *adds* received (23.6.11) by the Port's Receive Process.
- h) addConfirmationsReceived: The number of *add confirms* received (23.6.12) by the Port's Receive Process.
- i) lossNotificationsReceived: The number of *losses* received (23.6.13) by the Port's Receive Process.
- j) lossConfirmationsReceived: The number of *loss confirms* received (23.6.14) by the Port's Receive Process.
- k) addEvents: The number of transitions to STM:ADD directly from STM:DOWN or STM:LOSS (23.8).
- l) lossEvents: The number of transitions to STM:LOSS directly from STM:UP or STM:ADD (23.8).
- m) macStatusNotifications: The number of transitions to SNM:MAC_NOTIFICATION (23.9).

12.20 Management entities for forwarding and queueing for time-sensitive streams

The Bridge enhancements for support of forwarding and queuing for time-sensitive streams are defined in Clause 34.

The objects that comprise this managed resource are as follows:

- a) The Bandwidth Availability Parameter Table (12.20.1)
- b) The Transmission Selection Algorithm Table (12.20.2)
- c) The Priority Regeneration Override Table (12.20.3)

12.20.1 The Bandwidth Availability Parameter Table

There is one Bandwidth Availability Parameter Table per Port of a bridge component. Each table row contains a set of parameters for each traffic class that supports the credit-based shaper algorithm (8.6.8.2), as detailed in Table 12-1. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of ports and components.

Table 12-1—Bandwidth Availability Parameter Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
Traffic class	unsigned integer [0..7]	R	BE	34.3
deltaBandwidth	percentage	RW	BE	34.3
adminIdleSlope	unsigned integer	RW	BE	34.3
operIdleSlope	unsigned integer	R	BE	34.3

^aR = Read only access; RW = Read/Write access.

^bB = Required for bridge or bridge component support of forwarding and queueing for time-sensitive streams.

E = Required for end station support of forwarding and queueing for time-sensitive streams.

12.20.2 The Transmission Selection Algorithm Table

There is one Transmission Selection Algorithm Table per Port of a bridge component. This is in addition to the Traffic Class Table managed object (12.6.3) that would also exist per Port of a bridge component. Each table row contains a set of parameters for each traffic class that the Port supports, as detailed in Table 12-2.

Table 12-2—Transmission Selection Algorithm Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
Traffic class	unsigned integer [0..7]	R	B	8.6.8
Transmission selection algorithm	enumerated (see Table 8-5)	RW	B	8.6.8, Table 8-5

^aR = Read only access; RW = Read/Write access.

^bB = Required for bridge or bridge component support of forwarding and queueing for time-sensitive streams.

12.20.3 The Priority Regeneration Override Table

There is one Priority Regeneration Override Table per Port of a bridge component. This is in addition to the Priority Handling managed object (12.6.2) that would also exist per Port of a bridge component. Each table row contains a set of parameters for each priority value that is associated with an SR class (3.175, 6.9.4), as detailed in Table 12-3.

Table 12-3—Priority Regeneration Override Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
Received priority	integer [0..7]	R	B	6.9.4
Regenerated priority	integer [0..7]	RW	B	6.9.4
SRPdomainBoundaryPort	boolean	R	B	6.6.4

^aR = Read only access; RW = Read/Write access.

^bB = Required for bridge or bridge component support of forwarding and queueing for time-sensitive streams.

12.21 Congestion notification managed objects

A number of the variables that implement congestion notification, including Congestion Points (CPs), Reaction Points (RPs), and Congestion Notification Domain (CND) defense, are manageable objects.²⁶ There are a number of managed objects, each including a number of variables. The managed objects are as follows:

- a) CN component managed object (12.21.1);
- b) CN component priority managed object (12.21.2);
- c) CN Port priority managed object (12.21.3);
- d) Congestion Point managed object (12.21.4);
- e) Reaction Point port priority managed object (12.21.5); and
- f) Reaction Point group managed object (12.21.6).

NOTE—If multiple managed objects are altered over a period of time, then between the time the first and last object has been altered, operation of the state machines could produce unexpected results. This standard assumes that any number of managed objects can be altered as an atomic operation, so that no inconsistent intermediate states can occur. See 17.7.13 for one mechanism to ensure consistency.

12.21.1 CN component managed object

A single instance of the CN component managed object shall be implemented by a bridge component or end station that is congestion aware. It comprises all of the variables included in the CN component variables (32.2), as illustrated in Table 12-4. An end station may omit the managed objects noted “C” in the

²⁶The managed objects in this subclause are documented in a manner that is not parallel to that of the managed objects included in IEEE Std 802.1Q-2005 and certain other of its amendments. It is intended that managed object definitions in future amendments to IEEE Std 802.1Q will follow the format in this subclause. The resultant inconsistencies will be resolved in a future edition of IEEE Std 802.1Q, insofar as this can be done without invalidating the MIB definitions in Clause 17, some of which depend on the format of Clause 12 used in IEEE Std 802.1Q-2005.

Conformance column of Table 12-4 if it does not support CPs, and may in any case omit the managed object marked “e.”

Table 12-4—CN component managed object row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
cngMasterEnable	Boolean	RW	BE	32.2.1
cngCnmTransmitPriority	unsigned integer [0..7]	R	BC	32.2.2
cngDiscardedFrames	counter	R	BC	32.2.3
cngErroredPortList	list	R	Be	32.2.4

^aR = Read only access;

RW = Read/Write access

^bB = Required for a Bridge or Bridge component that is congestion aware;

C = Required for an end station that implements one or more CPs

E = Required for an end station that is congestion aware

e = Optional for an end station that is congestion aware

12.21.2 CN component priority managed object

The CN component priority managed object contains the managed objects that control a single Congestion Notification Priority Value (CNPV) for all Ports in an end station or bridge component. It comprises all of the variables included in the Congestion notification per-CNPV variables (32.3), as illustrated in Table 12-5. In one common use case for congestion notification, every Port of a bridge or end station has the same number of CPs, each configured for the same set of priority values. This managed object facilitates configuring that case. These objects can override, and can be overridden by, the CN Port priority managed objects (12.21.3) as described in 32.1.3.

Table 12-5—CN component priority managed object row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
cncpDefModeChoice	enum {cpcAdmin, cpcAuto}	RW	BE	32.3.1
cncpAlternatePriority	integer [0..7]	RW	BC	32.3.2
cncpAutoAltPri	integer [0..7]	R	BC	32.3.3
cncpAdminDefenseMode	enum {cptDisabled, cptInterior, cptInteriorReady, cptEdge}	RW	BE	32.3.4
cncpCreation	enum {cncpAutoEnable, cncpAutoDisable}	RW	BE	32.3.5
cncpLldpInstanceChoice	enum {cnlNone, cnlAdmin}	RW	BE	32.3.6
cncpLldpInstanceSelector	IEEE 802.1AB LLDP instance selector	RW	BE	32.3.7

^aR = Read only access;

RW = Read/Write access

^bB = Required for a Bridge or Bridge component that is congestion aware;

C = Required for an end station that implements one or more CPs

E = Required for an end station that is congestion aware

A CN component priority managed object shall be implemented by a bridge component or end station that is congestion aware for each priority value that can be a CNPV. An end station may omit the managed objects noted “C” in the Conformance column in Table 12-5 if it does not support Congestion Points (CPs). The operations that can be performed on a congestion-aware system’s CN component priority managed object are as follows:

- a) Create CN component priority managed object (12.21.2.1); and
- b) Delete CN component priority managed object (12.21.2.2).

12.21.2.1 Create CN component priority managed object

Creating a CN component priority managed object creates an instance of each of the Congestion notification per-CNPV variables (32.3), and also creates the corresponding CN Port priority managed objects (12.21.3) and all their dependent managed objects and variables, on every Port in the bridge component or end station, as illustrated in CN Port priority managed object. Depending on the value of cncpCreation (32.3.5), creating a CN component priority managed object can make the selected priority a CNPV throughout the bridge component or end station.

12.21.2.2 Delete CN component priority managed object

Deleting a CN component priority managed object deletes all of the CN component variables (32.2), and also deletes the corresponding CN Port priority managed objects (12.21.3) and all their dependent managed objects and variables, on all Ports in the bridge component or end station, thus making the priority not a CNPV throughout the bridge component or end station.

12.21.3 CN Port priority managed object

There is one CN Port priority managed object per Port per priority in a congestion aware end station or bridge component. It comprises some of the variables included in the CND defense per-Port per-CNPV variables (32.4), as illustrated in Table 12-6. These objects can override, and can be overridden by, the CN component managed objects (12.21.1) and CN component priority managed objects (12.21.2) as described in 32.1.3.

Table 12-6—CN Port priority managed object row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
cnpdDefModeChoice	enum {cpcAdmin, cpcAuto, cpcComp}	RW	BE	32.4.1
cnpdAdminDefenseMode	enum {cptDisabled, cptInterior, cptInteriorReady, cptEdge}	RW	BE	32.4.2
cnpdAutoDefenseMode	enum {cptDisabled, cptInterior, cptInteriorReady, cptEdge}	R	BE	32.4.3
cnpdLldpInstanceChoice	enum {cnlNone, cnlAdmin}	RW	BE	32.4.4
cnpdLldpInstanceSelector	IEEE 802.1AB LLDP instance selector	RW	BE	32.4.5
cnpdAlternatePriority	integer [0..7]	RW	BC	32.4.6

^aR = Read only access;

RW = Read/Write access

^bB = Required for a Bridge or Bridge component that is congestion aware;

C = Required for an end station that implements one or more CPs

E = Required for an end station that is congestion aware

A CN Port priority managed object is created or deleted when the corresponding Port or CN component priority managed object is created or deleted, and its initial state on creation is determined by cncpCreation (32.3.5).

The CN Port priority managed object shall be implemented by a bridge component or end station that is congestion aware. An end station may omit the managed objects noted “C” in the Conformance column in Table 12-6 if it does not support CPs.

12.21.4 Congestion Point managed object

There is one Congestion Point managed object for each CP in a bridge component or end station. It comprises some of the variables included in the Congestion Point variables (32.8), as illustrated in Table 12-7.

Table 12-7—Congestion Point managed object row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
cpMacAddress	MAC address	R	BC	32.8.1
cpId	octet string (size = 8)	R	BC	32.8.2
cpQSp	unsigned integer	R	BC	32.8.3
cpW	real number	RW	BC	32.8.6
cpSampleBase	unsigned integer	RW	BC	32.8.11
cpDiscardedFrames	counter	R	BC	32.8.12
cpTransmittedFrames	counter	R	BC	32.8.13
cpTransmittedCnms	counter	R	BC	32.8.14
cpMinHeaderOctets	unsigned integer	RW	BC	32.8.15

^aR = Read only access;

RW = Read/Write access

^bB = Required for a Bridge or Bridge component that is congestion aware;

C = Required for an end station that implements one or more CPs

The Congestion Point managed object shall be implemented by a congestion aware bridge component. It shall be implemented by an end station that supports CPs.

NOTE—The Recommended priority to traffic class mappings in Table 8-4 can assign more than one CNPV to the same traffic class, the same queue, and hence the same CP. There can be only one CP controlling a given queue, and that CP has one set of controlling managed objects, not one set per CNPV. That set of managed objects can be accessed using any of the CNPV values assigned to the CP’s queue. Thus, changing a managed object for one CNPV changes the managed object for all CNPVs assigned to the same queue.

12.21.5 Reaction Point port priority managed object

A congestion aware end station shall implement one Reaction Point port priority managed object for each CNPV on each port. The Reaction Point port priority managed object controls the creation of RPs on the port and CNPV, as illustrated in Table 12-8.

Table 12-8—Reaction Point port priority managed object row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
rpppMaxRps	unsigned integer	RW	E	32.10.1
rpppCreatedRps	counter	R	E	32.10.2
rpppRpCentiseconds	unsigned integer	R	E	32.10.3

^aR = Read only access;

RW = Read/Write access

^bE = Required for an end station that is congestion aware.

12.21.6 Reaction Point group managed object

There is one Reaction Point group managed object for each set of Reaction Point group variables (32.11), as illustrated in Table 12-9. The Reaction Point group managed object shall be implemented by an end station that is congestion aware.

Table 12-9—Reaction Point group managed object row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
rpgEnable	Boolean	RW	E	32.11.1
rpgTimeReset	unsigned integer	RW	E	32.11.2
rpgByteReset	unsigned integer	RW	E	32.11.3
rpgThreshold	unsigned integer	RW	E	32.11.4
rpgMaxRate	unsigned integer	RW	E	32.11.5
rpgAiRate	unsigned integer	RW	E	32.11.6
rpgHaiRate	unsigned integer	RW	E	32.11.7
rpgGd	real number	RW	E	32.11.8
rpgMinDecFac	real number	RW	E	32.11.9
rpgMinRate	unsigned integer	RW	E	32.11.10

^aRW = Read/Write access^bE = Required for an end station that is congestion aware

12.22 SRP entities

The Bridge enhancements for support of SRP (Stream Reservation Protocol) are defined in Clause 35.

The objects that comprise this managed resource are

- a) SRP Bridge Base Table (12.22.1)
- b) SRP Bridge Port Table (12.22.2)
- c) SRP Latency Parameter Table (12.22.3)

- d) SRP Stream Table (12.22.4)
- e) SRP Reservations Table (12.22.5)

12.22.1 SRP Bridge Base Table

There is a set of parameters that configure SRP operation for the entire device. Those parameters are shown in Table 12-10.

Table 12-10—SRP Bridge Base Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
msrpEnabledStatus	Boolean	RW	B	35.2.1.4(d)
talkerPruning	Boolean	RW	B	35.2.1.4(b)
msrpMaxFanInPorts	unsigned integer	RW	B	35.2.1.4(f)
msrpLatencyMaxFrameSize	unsigned integer	RW	B	35.2.1.4(g)

^aR = Read only access; RW = Read/Write access

^bB = required for bridge or bridge component support of SRP, E = required for end station support of SRP

12.22.2 SRP Bridge Port Table

There is one SRP Configuration Parameter Table per Port of a bridge component. Each table row contains a set of parameters for each MSRP entity per port, as detailed in Table 12-11.

Table 12-11—SRP Bridge Port Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
msrpPortEnabledStatus	Boolean	RW	B	35.2.1.4(e)
MSRP Failed Registrations	counter	R	B	10.7.12.1
MSRP Last PDU Origin	MAC Address	R	B	10.7.12.2
SR_PVID	unsigned integer[1..4094]	RW	B	Table 9-2 35.2.1.4(i)

^aR = Read only access; RW = Read/Write access

^bB = required for bridge or bridge component support of SRP, E = required for end station support of SRP

12.22.3 SRP Latency Parameter Table

There is one SRP Latency Parameter Table per Port of a bridge component. Each table row contains a set of parameters for each traffic class supported on a port, as detailed in Table 12-12. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of ports and components.

Table 12-12—SRP Latency Parameter Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
Traffic class	unsigned integer [0..7]	R	BE	34.3
portTcMaxLatency	unsigned integer	R	BE	35.2.1.4(a), 35.2.2.8.6

^aR = Read only access; RW = Read/Write access^bB = required for bridge or bridge component support of SRP, E = required for end station support of SRP

12.22.4 SRP Stream Table

There is one SRP Stream Table per bridge component. Each table contains a set of parameters for each StreamID that is registered on the Bridge, as detailed in Table 12-13. Rows in the table are created and removed dynamically as StreamIDs are registered and deregistered on the Bridge.

Table 12-13—SRP Stream Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
StreamID	octet string(size(8))	RW	BE	35.2.2.8.2
Stream Destination Address	MAC Address	R	BE	35.2.2.8.3(a)
Stream VLAN ID	unsigned integer [0..4094]	R	BE	35.2.2.8.3(b)
MaxFrameSize	unsigned integer [0..65535]	R	BE	35.2.2.8.4(a)
MaxIntervalFrames	unsigned integer [0..65535]	R	BE	35.2.2.8.4(b)
Data Frame Priority	unsigned integer [0..7]	R	BE	35.2.2.8.5(a)
Rank	unsigned integer [0..1]	R	BE	35.2.2.8.5(b)

^aR = Read only access; RW = Read/Write access^bB = required for bridge or bridge component support of SRP, E = required for end station support of SRP

12.22.5 SRP Reservations Table

There is one SRP Reservations Table per reservation direction per port of a bridge component. Each table contains a set of parameters for each Talker or Listener Reservation that is registered on a port of the Bridge, as detailed in Table 12-14. Rows in the table can be created or removed dynamically as Talker and Listener declarations are registered and deregistered on a port of the Bridge.

Table 12-14—SRP Reservations Table row elements

Name	Data type	Operations supported ^a	Conformance ^b	References
StreamID	octet string(size(8))	RW	BE	35.2.2.8.2
Direction	unsigned integer[0..1]	R	BE	35.2.1.2
Declaration Type	unsigned integer [0..4]	R	BE	35.2.1.3
Accumulated Latency	unsigned integer	R	BE	35.2.2.8.6
Failed Bridge ID	BridgeId	R	BE	35.2.2.8.7(a)
Failure Code	unsigned integer [0..16]	R	BE	35.2.2.8.7(b)
Dropped Frames	counter	R	BE	35.2.5.1
Stream Age	unsigned integer	R	BE	35.2.1.4(c)

^aR = Read only access; RW = Read/Write access^bB = required for bridge or bridge component support of SRP, E = required for end station support of SRP

13. Spanning Tree Protocols

The spanning tree algorithms and protocols specified by this standard provide simple and full connectivity throughout a Bridged Local Area Network comprising arbitrarily interconnected bridges. Each bridge can use the Rapid Spanning Tree Protocol (RSTP), or the Multiple Spanning Tree Protocol (MSTP).

NOTE 1—The spanning tree protocols specified by this standard supersede the Spanning Tree Protocol (STP) specified in IEEE Std 802.1D revisions prior to 2004, but facilitate migration by interoperating with the latter without configuration restrictions beyond those previously imposed by STP. However networks that include bridges using STP can reconfigure slowly and constrain active topologies.

NOTE 2—Although the active topologies determined by the spanning tree protocols connect all the components of a Bridged Local Area Network, filtering (MVRP, etc.) can restrict frames to a subset of each active topology.

RSTP assigns all frames to a Common Spanning Tree (CST), without being aware of the active topology assignments made by MSTP that allow frames to follow separate paths within Multiple Spanning Tree (MST) Regions. Each of these regions comprises MST Bridges that consistently assign any given frame to the same active topology (see 8.4) and the LANs that interconnect those bridges. These regions and other bridges and LANs are connected into the CST, to provide loop-free network wide connectivity even if active topology assignments or spanning tree protocols differ locally.

MSTP connects all bridges and LANs with a single Common and Internal Spanning Tree (CIST) that supports the automatic determination of each region, choosing its maximum possible extent. The connectivity calculated for the CIST provides the CST for interconnecting the regions, and an Internal Spanning Tree (IST) within each region. MSTP calculates a number of independent Multiple Spanning Tree Instances (MSTIs) within each region, and ensures that frames with a given VID are assigned to one and only one of the MSTIs or the IST within the region (or reserved for use by PBB-TE), that the assignment is consistent among all bridges within the region, and that the stable connectivity of each MSTI and the IST at the boundary of the region matches that of the CST. Spanning tree protocol entities transmit and receive BPDUs (Clause 14) to convey parameters used by RSTP and MSTP to calculate CST, CIST, and MSTI spanning trees. BPDUs also convey parameters that all the spanning tree protocols use to interoperate with each other, that determine the extent of MST Regions, and that ensure that temporary loops are not created when neighboring bridges are acting on different topology information.

This clause

- a) Specifies protocol design and support requirements (13.1, 13.2) and design goals (13.3).
- b) Provides an overview of RSTP (13.4) and MSTP (13.5) operation.
- c) Describes how the spanning tree protocols interoperate and coexist (13.6).
- d) Specifies how spanning tree priority vectors (13.8) are calculated (13.9, 13.10) and used to assign the Port Roles (13.11) that determine the Port States, i.e. forwarding and learning (8.4), for each tree.
- e) Shows that RSTP and MSTP provide stable connectivity (13.12).
- f) Describes how spanning tree priority vectors are communicated (13.13) and changed (13.14).
- g) Describes how Port Roles are used to change Port States without introducing loops (13.15, 13.16).
- h) Recommends defaults and ranges for the parameters that determine each tree's topology (13.16).
- i) Describes the updating of learned station location information when a tree reconfigures (13.17).
- j) Specifies additional controls that can speed reconfiguration or prevent unwanted outcomes (13.18).
- k) Describes how loops are prevented when a LAN is only providing one-way connectivity (13.19), and can be prevented when the network includes bridges whose protocol operation can fail (13.21).
- l) Describes how a bridge's protocol processing can be 'hot upgraded' in an active network (13.20).
- m) Specifies RSTP and MSTP using state machines (13.22–13.38).
- n) Specifies the use and configuration of the spanning tree protocols for the special cases of a Provider Edge Bridge's Customer Edge Ports (13.39), a Backbone Edge Bridge's Virtual Instance Ports (13.40), and an L2 Gateway Port connecting a customer to a provider (13.40).

NOTE 3—Readers of this specification are urged to begin by familiarizing themselves with RSTP.

Clause 14 specifies the format of BPDUs. The text of this clause (Clause 13) takes precedence should any conflict be apparent between it and the text in other parts of this standard (in particular, Clause 12, Clause 14, and Annex A). Within this clause (Clause 13) the state machine specifications (13.22–13.37) takes precedence over the general description (13.1–13.21). A distinctive font is used to highlight references to state machine variables, procedures, and STATES in the general description.

13.1 Protocol design requirements

The Spanning Tree Algorithm and its associated protocols operate in Bridged Local Area Networks of arbitrary physical topology comprising MST or SST Bridges connecting shared media or point-to-point LANs, so as to support, preserve, and maintain the quality of the MAC Service in all its aspects as specified by Clause 6.

RSTP configures the Port State (8.4) of each Bridge Port. MSTP configures the Port State for the CIST and each MSTI, and verifies the allocation of VIDs to FIDs and FIDs to trees. Operating both independently and together, RSTP and MSTP meet the following requirements:

- a) They configure one or more active topologies that fully connect all physically connected LANs and bridges, and stabilize (with high probability) within a short, known bounded interval after any change in the physical topology, maximising service availability (6.5.1).
- b) The active topology for any given frame remains simply connected at all times (6.5.3, 6.5.4), and will (with high probability) continue to provide simple and full connectivity for frames even in the presence of administrative errors (e.g. in the allocation of VIDs to MSTIs).
- c) The configured stable active topologies are unicast multicast congruent, downstream congruent, and reverse path congruent (symmetric) (3.114, 3.193, 6.3).
- d) The same symmetric active topology is used, in a stable network, for all frames using the same FID, i.e. between any two LANs all such frames are forwarded through the same Bridge Ports (6.3).
- e) The active topology for a given VID can be chosen by the network administrator to be a common spanning tree, or one of multiple spanning trees (if MSTP is implemented).
- f) Each active topology is predictable, reproducible, and manageable, allowing Configuration Management (following traffic analysis) to meet Performance Management goals (6.5 and 6.5.10).
- g) The configured network can support VLAN-unaware end stations, such that they are unaware of their attachment to a single LAN or a Bridged Local Area Network, or their use of a VID (6.2).
- h) The communications bandwidth on any particular LAN is always a small fraction of the total available bandwidth (6.5.10).

NOTE—The spanning tree protocols cannot protect against temporary loops caused by the interconnection of LANs by devices other than bridges (e.g., LAN repeaters) that operate invisibly with respect to support of the MAC Internal Sublayer Service and the MAC_Operational status parameter (6.6.2).

13.2 Protocol support requirements

In order for the spanning tree protocols to operate, the following are required:

- a) A unique Group MAC Address used by the Spanning Tree Protocol Entities (8.10) of participating bridges or bridge components (5.2), and recognized by all the bridges attached to a LAN.
- b) An identifier for each bridge or bridge component, unique within the Bridged Local Area Network.
- c) An identifier for each Bridge Port, unique within a bridge or bridge component.

Values for each of these parameters shall be provided by each bridge. The unique MAC Address that identifies the Spanning Tree Protocol Entities of MAC Bridges, VLAN Bridges (5.9), and C-VLAN components (5.5) is the Bridge Group Address (Table 8-1). The unique MAC Address that identifies the Spanning Tree Protocol Entities of S-VLAN components is the Provider Bridge Group Address (Table 8-2).

To allow management of active topology (for RSTP or MSTP) means of assigning values to the following are required:

- d) The relative Bridge Priority of each bridge in the network.
- e) A Port Path Cost for each Bridge Port.
- f) The relative Port Priority of each Bridge Port.

13.2.1 MSTP support requirements

MSTP does not require any additional configuration, provided that communication between end stations is supported by a number of VLANs. However, to realize the improved throughput and associated frame loss and transit delay performance improvements made possible by the use of multiple spanning trees, the following are required:

- a) Assessment of the probable distribution of traffic between VLANs and between sets of communicating end stations using those VLANs.
- b) Per MSTI assignment of Bridge Priority and Internal Port Path Costs to configure the MSTIs.
- c) Consistent assignment of VIDs to MSTIDs within each potential MST Region (3.113).
- d) Administrative agreement on the Configuration Name and Revision Level used to represent the assignments of VIDs to MSTIDs.

13.3 Protocol design goals

All the spanning tree protocols meet the following goal, which simplifies operational practice:

- a) Bridges do not have to be individually configured before being added to the network, other than having their MAC Addresses assigned through normal procedures.
- b) In normal operation, the time taken to configure the active topology of a network comprising point-to-point LANs is independent of the timer values of the protocol.

RSTP and MSTP meet the following goal, which limits the complexity of bridges and their configuration:

- c) The memory requirements associated with each Bridge Port are independent of the number of bridges and LANs in the network.

13.4 RSTP overview

The Rapid Spanning Tree Protocol (RSTP) configures the Port State (8.4) of each Bridge Port in the Bridge Local Area Network. RSTP ensures that the stable connectivity provided by each bridge between its ports and by the individual LANs attached to those ports is predictable, manageable, full, simple, and symmetric. RSTP further ensures that temporary loops in the active topology do not occur if the network has to reconfigure in response to the failure, removal, or addition of a network component, and that erroneous station location information is removed from the Filtering Database after reconfiguration.

Each of the bridges in the network transmits Configuration Messages (13.13). Each message contains spanning tree priority vector (13.8) information that identifies one bridge as the Root Bridge of the network, and allows each bridge to compute its own lowest path cost to that Root Bridge before transmitting its own Configuration Messages. A Port Role (13.11) of Root Port is assigned to the one port on each bridge that provides that lowest cost path to the Root Bridge, and a Port Role of Designated Port to the one Bridge Port that provides the lowest cost path from the attached LAN to the Root Bridge. Alternate Port and Backup Port roles are assigned to Bridge Ports that can provide connectivity if other network components fail.

State machines associated with the Port Roles maintain and change the Port States that control forwarding (8.6) and learning (8.7) of frames. In a stable network, Root Ports and Designated Ports are Forwarding, while Alternate, Backup, and Disabled Ports are Discarding. Each Port's role can change if a Bridge, Bridge Port, or LAN fails, is added to, or removed from network. Port state transitions to Learning and Forwarding are delayed, and ports can temporarily transition to the Discarding state to prevent loops and to ensure that misordering (6.5.3) and duplication (6.5.4) rates remain negligible.

RSTP provides rapid recovery of connectivity to minimize frame loss (6.5.2). A new Root Port, and Designated Ports attached to point-to-point LANs, can transition to Forwarding without waiting for protocol timers to expire. A Root Port can transition to Forwarding without transmitting or receiving messages from other bridges, while a Designated Port attached to a point-to-point LAN can transition when it receives an explicit agreement transmitted by the other bridge attached to that LAN. The forwarding transition delay used by a Designated Port attached to a shared media LAN is long enough for other bridges attached to that LAN to receive and act on transmitted messages, but is independent of the overall network size. If all the LANs in a network are point-to-point, RSTP timers define worst-case delays that only occur if protocol messages are lost or rate transmission limits are exceeded.

A Bridge Port attached to a LAN that has no other bridges attached to it may be administratively configured as an Edge Port. RSTP monitors the LAN to ensure that no other bridges are connected, and may be configured to automatically detect an Edge Port. Each Edge Port transitions directly to the Forwarding Port State, since there is no possibility of it participating in a loop.

13.4.1 Computation of the active topology

The bridge with the best Bridge Identifier is selected as the Root Bridge. The unique Bridge Identifier for each bridge is derived, in part, from the Bridge Address (8.13.8) and, in part, from a manageable priority component (13.24). The relative priority of bridges is determined by the numerical comparison of the unique identifiers, with the lower numerical value indicating the better identifier.

Every bridge has a Root Path Cost associated with it. For the Root Bridge this is zero. For all other bridges, it is the sum of the Port Path Costs on the least cost path to the Root Bridge. Each port's Path Cost may be managed, 13.16 recommends default values for ports attached to LANs of various speeds.

The Bridge Port on each bridge with the lowest Root Path Cost is assigned the role of Root Port for that bridge (the Root Bridge does not have a Root Port). If a bridge has two or more ports with the same Root Path Cost, then the port with the best Port Identifier is selected as the Root Port. Part of the Port Identifier is fixed and different for each port on a bridge, and part is a manageable priority component (13.24). The relative priority of Bridge Ports is determined by the numerical comparison of the unique identifiers, with the lower numerical value indicating the better identifier.

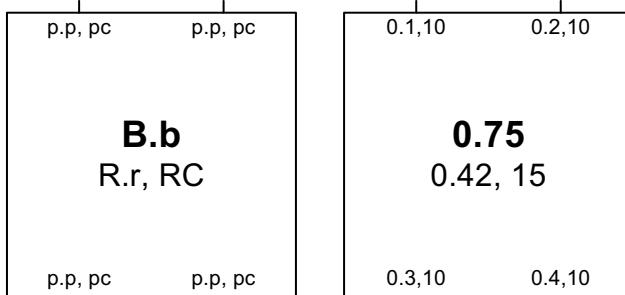
Each LAN in the Bridged Local Area Network also has an associated Root Path Cost. This is the Root Path Cost of the lowest cost bridge with a Bridge Port connected to that LAN. This bridge is selected as the Designated Bridge for that LAN. If there are two or more bridges with the same Root Path Cost, then the bridge with the best priority (least numerical value) is selected as the Designated Bridge. The Bridge Port on the Designated Bridge that is connected to the LAN is assigned the role of Designated Port for that LAN. If the Designated Bridge has two or more ports connected to the LAN, then the Bridge Port with the best priority Port Identifier (least numerical value) is selected as the Designated Port.

In a Bridged Local Area Network whose physical topology is stable, i.e RSTP has communicated consistent information throughout the network, every LAN has one and only one Designated Port, and every bridge with the exception of the Root Bridge has a single Root Port connected to a LAN. Since each bridge provides connectivity between its Root Port and its Designated Ports, the resulting active topology connects all LANs (is “spanning”) and will be loop-free (is a “tree”).

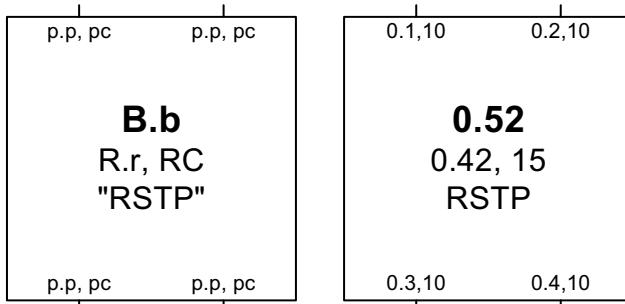
Any Bridge Port that is enabled, but not a Root or Designated Port, is a Backup Port if that bridge is the Designated Bridge for the attached LAN, and an Alternate Port otherwise. An Alternate Port offers an alternate path in the direction of the Root Bridge to that provided by the bridge’s own Root Port, whereas a Backup Port acts as a backup for the path provided by a Designated Port in the direction of the leaves of the Spanning Tree. Backup Ports exist only where there are two or more connections from a given bridge to a given LAN; hence, they (and the Designated Ports that they back up) can only exist where the bridge has two or more ports attached to a shared media LAN, or directly connected by a point-to-point LAN.

13.4.2 Example topologies

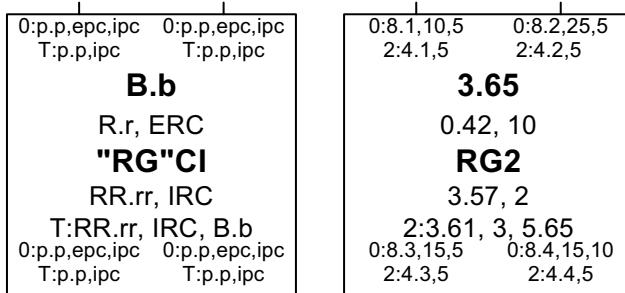
The spanning tree examples in this clause use the conventions of Figure 13-1.



A template for and example of an STP Bridge. **B.b** is the Bridge Identifier (including the manageable priority component **B**). **R.r** and **RC** are the Root Identifier, Root Path Cost, and the Designated Bridge Identifier, for the Root Port. **p.p, pc** are the Port Identifier (with manageable priority **p**) and the Port Path Cost for a Bridge Port.



A template for and example of an RSTP Bridge.



A template for and an example of an MSTP Bridge. **B.b** is the CIST Bridge Identifier. **R.r, ERC**, **RR.rr** are the CIST Root Identifier, External Root Path Cost, and Regional Root Identifier. **CI** identifies the Configuration Identifier for the Bridge. **RR.rr, IRC** the CIST Regional Root Identifier and the Internal Root Path Cost. **T:RR.rr, IRC, B.b**, is the Regional Root Identifier, Internal Root Path Cost **IRC**, and Bridge Identifier for the MSTI with MSTID **T**.

0:p.p, epc, ipc are the CIST Port Identifier, External Port Path Cost, and Internal Port Path Cost for a Bridge Port. **T:p.p, ipc** are the Port Identifiers and their Regional Costs for MSTI **T**.

Any of the above information may be selectively omitted if deemed irrelevant for the purposes of a diagram.



A LAN

Connections between Bridges and LANs indicate the Port Role and Port State by means of their end point symbols, and in some examples, may show the transmission of BPDUs from a Port onto a LAN by means of arrowheads, as shown in the following table.

Port Role	Port State	Legend
Designated	Discarding Learning Forwarding	● — ● — ●—
& operEdge	Forwarding	●◇—
Root Port or Master Port	Discarding Learning Forwarding	O — O — O—
Alternate	Discarding Learning Forwarding	— — — — ——
Backup	Discarding Learning Forwarding	—III— —>I— —>—
Disabled	-	—X—
Transmitted Bpdus		
Designated		→
Designated Proposal		→→
Root		→→→
Root Agreement		→→→→

NOTE—These diagrammatic conventions allow the representation of Alternate and Backup Ports that are in Learning or Forwarding states; this can happen as a transitory condition due to implementation-dependent delays in switching off Learning and/or Forwarding on a Port that changes role from Designated or Root to Alternate or Backup.

Figure 13-1—Diagrammatic conventions for spanning tree topologies

Figure 13-2 shows a simple, redundantly connected, structured wiring configuration, with bridges connected by point-to-point LANs A through N, and a possible spanning tree active topology of the same network. Bridge 111 has been selected as the Root (though one cannot tell simply by looking at the active topology which bridge is the Root).

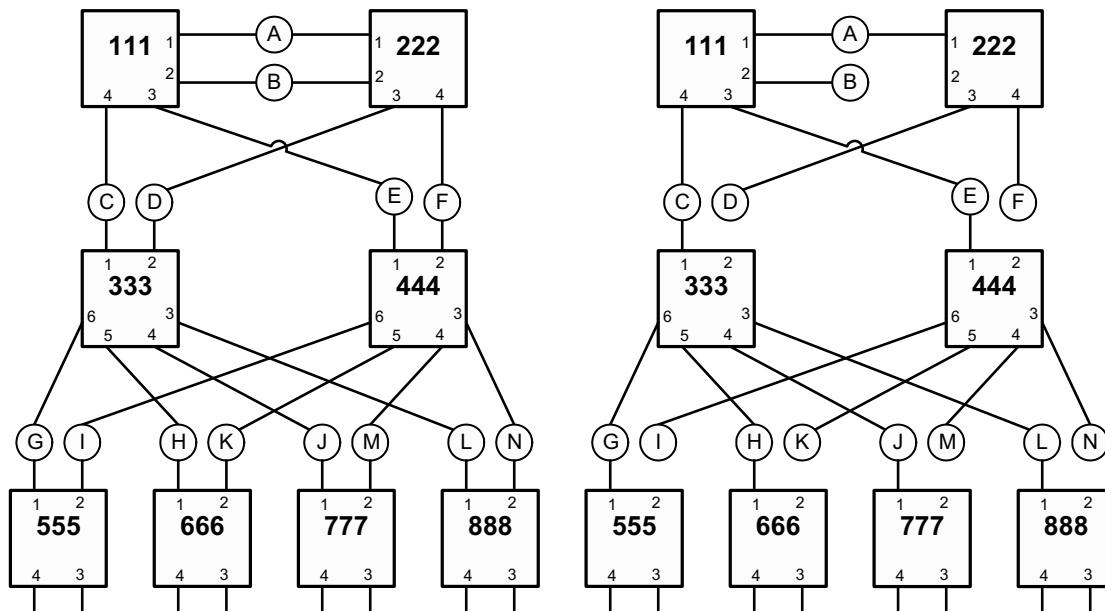


Figure 13-2—Physical topology and active topology

Figure 13-3 shows the Port Roles and Port States of each Bridge Port. It can be seen that bridge 111 is the Root, as its Ports are all Designated Ports, each of the remaining bridges have one Root Port.

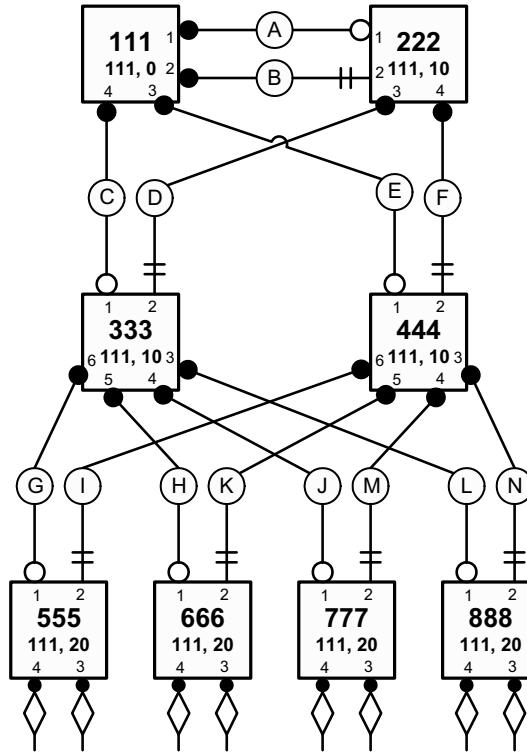


Figure 13-3—Port Roles and Port States

Figure 13-4 shows the result of connecting two of the ports of bridge 888 to the same LAN. As port 4 of bridge 888 has worse priority than port 3 and both offer the same Root Path Cost, port 4 will be assigned the Backup Port Role and will therefore be in the Discarding Port State. Should port 3 or its connection to LAN O fail, port 4 will be assigned the Designated Port Role and will transition to the Forwarding Port State.

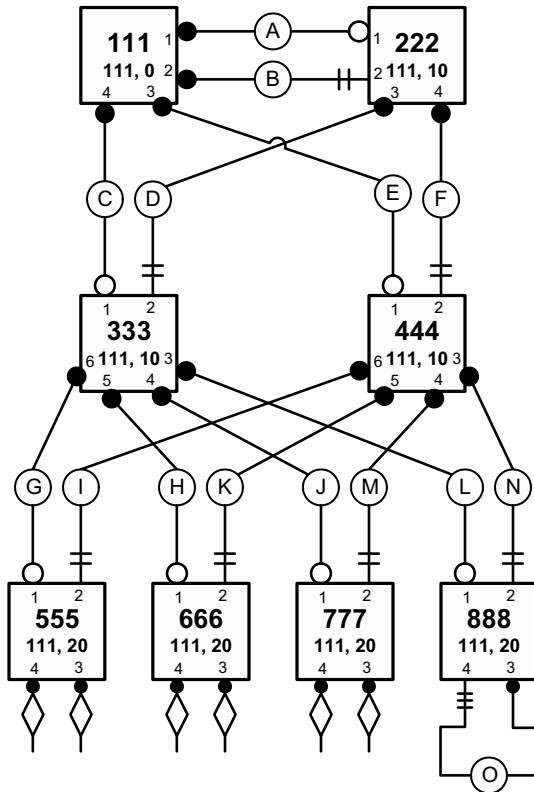


Figure 13-4—A Backup Port

Figure 13-5 shows a “ring” topology constructed from point-to-point links, as in some resilient backbone configurations. Bridge 111 is the Root, as in previous examples.

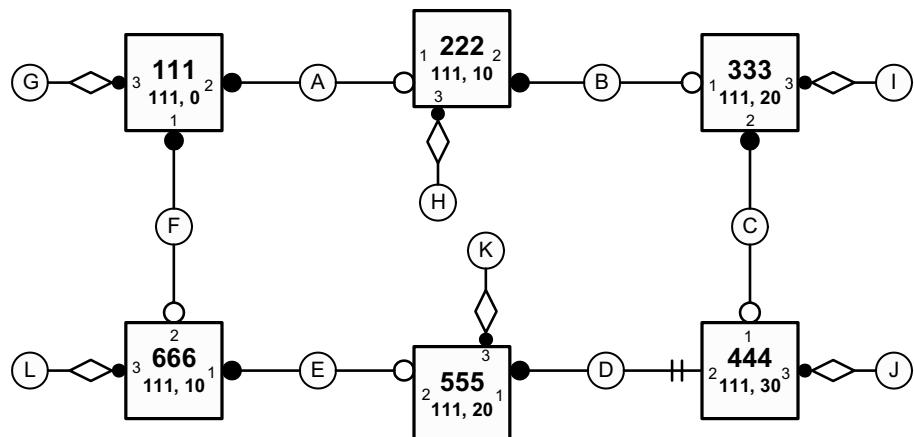


Figure 13-5—“Ring Backbone” example

13.5 MSTP overview

The Multiple Spanning Tree Protocol specifies:

- a) An MST Configuration Identifier (13.7) that allows each bridge to advertise its assignment, to a specified MSTI or to the IST, of frames with any given VID.
- b) A priority vector (13.8) that comprises bridge identifier and path cost information for constructing a deterministic and manageable single spanning tree active topology, the CIST, that:
 - 1) Fully and simply connects all bridges and LANs in a Bridged Local Area Network.
 - 2) Permits the construction and identification of MST Regions (3.113) of bridges and LANs that are guaranteed fully connected by the bridges and LANs within each region.
 - 3) Ensures that paths within each MST Region are always preferred to paths outside the region.
- c) An MSTI priority vector (13.8), comprising information for constructing a deterministic and independently manageable active topology for any given MSTI within each region.
- d) Comparisons and calculations performed by each bridge in support of the distributed spanning tree algorithm (13.9). These select a CIST priority vector for each Bridge Port, based on the priority vectors and MST Configuration Identifiers received from other bridges and on an incremental Path Cost associated with each reception Port. The resulting priority vectors are such that in a stable network:
 - 1) One bridge is selected to be the CIST Root of the Bridged Local Area Network as a whole.
 - 2) A minimum cost path to the CIST Root is selected for each bridge and LAN, thus preventing loops while ensuring full connectivity.
 - 3) The one bridge in each MST Region whose minimum cost path to the Root is not through another bridge using the same MST Configuration Identifier is identified as its region's CIST Regional Root.
 - 4) Conversely, each bridge whose minimum cost path to the Root is through a bridge using the same MST Configuration Identifier is identified as being in the same region as that bridge.
- e) Priority vector comparisons and calculations performed by each bridge for each MSTI (13.10). In a stable network:
 - 1) One bridge is independently selected for each MSTI to be the MSTI Regional Root.
 - 2) A minimum cost path to the MSTI Regional Root that lies wholly within the region is selected for each bridge and LAN.
- f) CIST Port Roles (13.11) that identify the role in the CIST active topology played by each port on a bridge.
 - 1) The Root Port provides the minimum cost path from the bridge to the CIST Root (if the bridge is not the CIST Root) through the Regional Root (if the bridge is not a Regional Root).
 - 2) A Designated Port provides the least cost path from the attached LAN through the bridge to the CIST Root.
 - 3) Alternate or Backup Ports provide connectivity if other bridges, Bridge Ports, or LANs fail or are removed.
- g) MSTI Port Roles (13.11) that identify the role played by each port on a bridge for each MSTI's active topology within and at the boundaries of a region.
 - 1) The Root Port provides the minimum cost path from the bridge to the Regional Root (if the bridge is not the Regional Root for the tree).
 - 2) A Designated Port provides the least cost path from the attached LAN though the bridge to the Regional Root.
 - 3) A Master Port provides connectivity from the region to a CIST Root that lies outside the region. The Bridge Port that is the CIST Root Port for the CIST Regional Root is the Master Port for all MSTIs.
 - 4) Alternate or Backup Ports provide connectivity if other bridges, Bridge Ports, or LANs fail or are removed.
- h) State machines and state variables associated with each spanning tree (CIST, or MSTI), port, and port role, to select and change the Port State (8.4, 13.22) that controls the processing and forwarding of frames assigned to that tree by a MAC Relay Entity (8.3).

13.5.1 Example topologies

Figure 13-6 is an example Bridged Local Area Network, using the conventions of Figure 13-1, and chosen to illustrate MSTP calculations rather than as an example of a common or desirable physical topology.

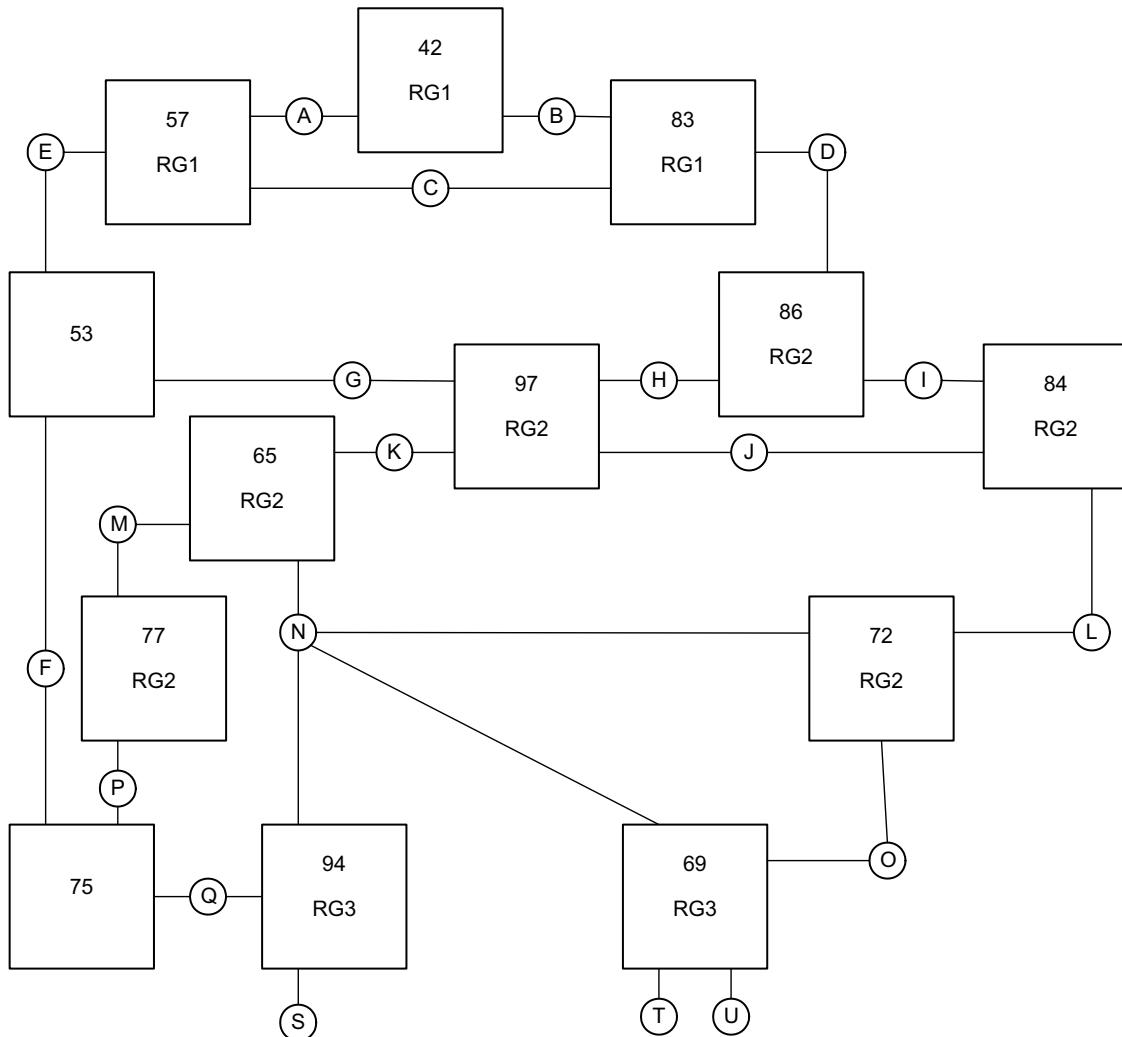


Figure 13-6—An MST Bridge network

Figure 13-7 is the same network showing bridges and LANs with better CIST spanning tree priorities higher on the page, and including CIST priority vectors, port roles, and MST Regions (3.113). In this example:

- Bridge 0.42 is the CIST Root because it has the best (numerically lowest) Bridge Identifier.
- Bridges 0.57 and 2.83 are in the same MST Region (1) as 0.42, because they have the same MST Configuration Identifier as the latter. Because they are in the same MST Region as the CIST Root, their External Root Path Cost is 0, and their CIST Regional Root is the CIST Root.
- LANs A, B, C, and D are in Region 1 because their CIST Designated Bridge is a Region 1 MST Bridge, and no STP bridges are attached to those LANs. LAN E is not in an MST Region (or in its own region—an equivalent view) because it is attached to bridge 0.53, which is not an MST Bridge.
- Bridges 0.77, 0.65, 0.97, 0.86, 3.84, and 3.72 are in the same MST Region (2) since they have the same MST Configuration Identifier and are interconnected by LANs for which one of them is the CIST Designated Bridge.

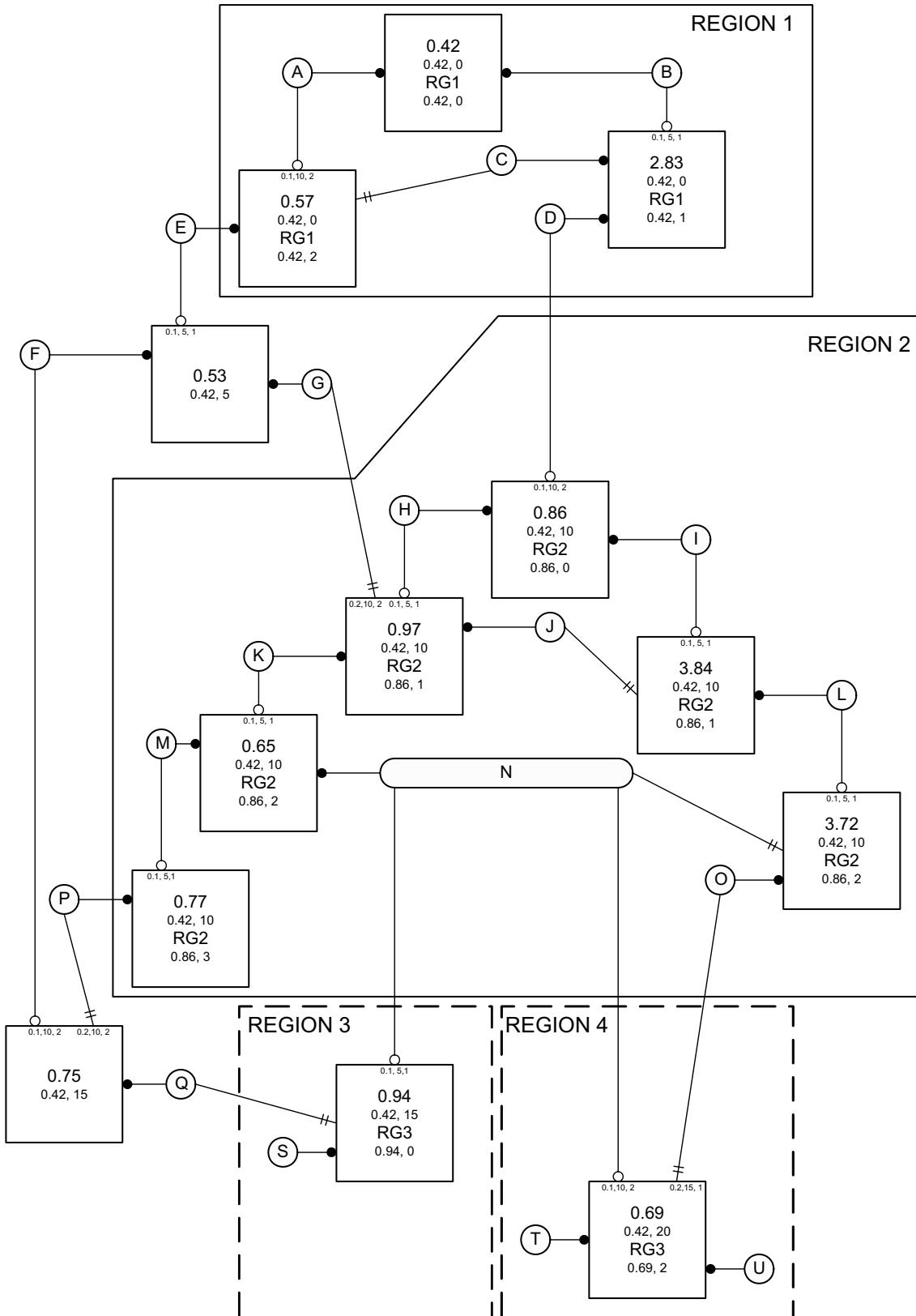


Figure 13-7—CIST Priority Vectors, Port Roles, and MST Regions

- e) Bridge 0.86 is the CIST Regional Root for Region 2 because it has the lowest External Root Path Cost through a Boundary Port.
- f) LAN N is in Region 2 because its CIST Designated Bridge is in Region 2. Frames assigned to different MSTIDs may reach N from bridge 0.86 (for example) by either bridge 0.65 or bridge 3.72, even though bridges 0.94 and 0.69 with MST Configuration Identifiers that differ from those for bridges in Region 2 are attached to this shared LAN.
- g) Bridges 0.94 and 0.69 are in different regions, even though they have the same MST Configuration Identifier, because the LAN that connects them (N) is in a different region.

Figure 13-8 shows a possible active topology of MSTI 2 within Region 2.

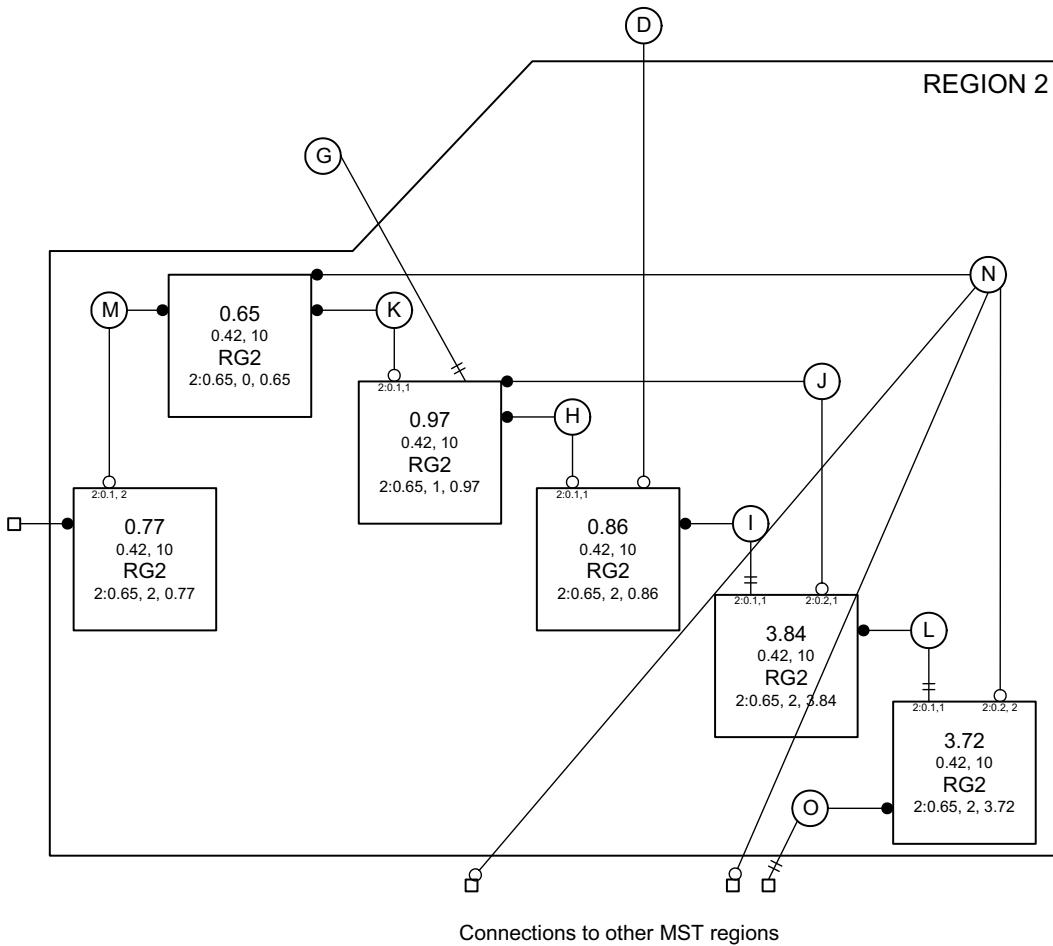


Figure 13-8—MSTI Active Topology in Region 2

- h) Bridge 0.65 has been chosen as the MSTI Regional Root because it has the best (numerically the lowest) Bridge Identifier of all bridges in the region for this MSTI.
- i) The connectivity between the whole of Region 2 and Region 1 is provided through a single Bridge Port, the Master Port on bridge 0.86. This port was selected for this role because it is the CIST Root Port on the CIST Regional Root for the region (see Figure 13-6).
- j) The connectivity between the whole of Region 2 and LANs and bridges outside the region for the MSTI is the same as that for the CIST. This connectivity is similar to that which might result by replacing the entire region by a single SST Bridge. The region has a single Root Port (this port is the Master Port for each MSTI) and a number of Designated Ports.

13.5.2 Relationship of MSTP to RSTP

MSTP is based on RSTP, extended so frames for different VLANs can follow different trees within regions.

- a) The same fundamental spanning tree algorithm selects the CIST Root Bridge and Port Roles, but extended priority vector components are used within (13.8, 13.9) in each region. As a result each region resembles a single bridge from the point of view of the CST as calculated by RSTP.
- b) Each MSTI's Regional Root Bridge and Port Roles are also computed using the same fundamental spanning tree algorithm with modified priority vector components (13.10).
- c) Different bridges may be selected as the Regional Root for different MSTIs by modifying the manageable priority component of the Bridge Identifier differently for the MSTIs.
- d) MST Configuration Identification is specific to MSTP.
- e) The Port Roles used by the CIST (Root, Designated, Alternate, Backup or Disabled Port) are the same as those of RSTP. The MSTIs use the additional port role Master Port. The Port States associated with each spanning tree and port are the same as those of RSTP.
- f) The state variables for each Bridge Port for each tree and for the bridge itself are those specified for RSTP as per port and per bridge with a few exceptions, additions, and enhancements.
- g) The performance parameters specified for RSTP apply to the CIST, with a few exceptions, additions, and enhancements. A simplified set of performance parameters apply to the MSTIs.
- h) This standard specifies RSTP state machines and procedures as a subset of MSTP.

13.5.3 Modeling an MST Region as a single bridge

The nominal replacement of an entire region by a single RSTP Bridge leads to little impact on the remainder of the Bridged Local Area Network. This design is intended to assist those familiar with RSTP to comprehend and verify MSTP, and to administer networks using MSTP. Treating the MST Regions as single bridges provides the network administrator with a natural hierarchy. The internal management of MST Regions (3.113) can be largely separated from the management of the active topology of the network as a whole.

The portion of the active topology of the network that connects any two bridges in the same MST Region traverses only MST Bridges and LANs in that region and never bridges of any kind outside the region; in other words, connectivity within the region is independent of external connectivity. This is because the protocol parameters that determine the active topology of the network as a whole, the Root Identifier and Root Path Cost (known in the MSTP specification as the CIST Root Identifier and CIST External Root Path Cost) are carried unchanged throughout and across the MST Region, so bridges within the region will always prefer spanning tree information that has been propagated within the region to information that has exited the region and is attempting to reenter it.

NOTE 1—No LAN can be in more than one MST Region at a time, so two bridges (0.11 and 0.22 say) that would otherwise be in the same region by virtue of having the same MST Configuration and of being directly connected by a LAN, may be in distinct regions if that is a shared LAN with other bridges attached (having a different MST Configuration) and no other connectivity between 0.11 and 0.22 and lying wholly within their region is available. The region that the shared LAN belongs to may be dynamically determined. No such dynamic partitioning concerns arise with single bridges. Obviously the sharing of LANs between administrative regions militates against the partitioning of concerns and should only be done following careful analysis.

The Port Path Cost (MSTP's External Port Path Cost) is added to the Root Path Cost just once at the Root Port of the CIST Regional Root, the closest bridge in the region to the Root Bridge of the entire network. The Message Age used by STP and RSTP is also only incremented at this port. If the CIST Root is within a region, it also acts as the Regional Root, and the Root Path Cost and Message Age advertised are zero, just as for a single bridge.

Within an MST Region, each MSTI operates in much the same way as an independent instance of RSTP with dedicated Regional Root Identifier, Internal Root Path Cost, and Internal Port Path Cost parameters.

Moreover, the overall spanning tree (the CIST) includes a fragment (the IST) within each MST Region that can be viewed as operating in the same way as an MSTI with the Regional Root as its root.

NOTE 2—Since an MST Region behaves like a single bridge and does not partition (except in the unusual configuration involving shared LANs noted above), it has a single Root Port in the CST active topology. Partitioning a network into two or more regions can therefore force nonoptimal blocking of Bridge Ports at the boundaries of those regions.

13.6 Compatibility and interoperability

RSTP and MSTP are designed to interoperate with each other and with STP. This clause (13.6) reviews aspects of their design that are important to meeting that requirement.

13.6.1 Designated Port selection

Correct operation of the spanning tree protocols requires that all Bridge Ports attached to any given LAN agree on a single CIST Designated Port after a short interval sufficient for any Bridge Port to receive a configuration message from that Designated Port.

A unique spanning tree priority (13.8) is required for each Bridge Port for STP, which has no other way of communicating port roles. Since port numbers on different bridges are not guaranteed to be unique, this necessitates the inclusion of the transmitting bridge's Bridge Identifier in the STP BPDU. RSTP and MSTP's Port Protocol Migration state machines (13.30) ensure that all bridges attached to any LAN with an attached STP bridge send and receive STP BPDUs exclusively.

NOTE 1—This behavior satisfies the requirement for unique, agreed Designated Port for LANs with attached STP bridges, but means that an MST Region cannot completely emulate a single bridge since the transmitted Designated Bridge Identifier can differ on Bridge Ports at the region's boundary.

MSTP transmits and receives the Regional Root Identifier and not the Designated Bridge Identifier in the BPDU fields recognized by RSTP (14.6) to allow both the MSTP and the RSTP Bridges potentially connected to a single LAN to perform comparisons (13.8, 13.9) between all spanning tree priority vectors transmitted that yield a single conclusion as to which RSTP Bridge or MST Region includes the Designated Port. MST and RST BPDUs convey the transmitting port's CIST Port Role. This is checked on receipt by RSTP when receiving messages from a Designated Bridge, thus ensuring that an RSTP Bridge does not incorrectly identify one MST Bridge Port as being Designated rather than another, even while omitting the competing Bridge Ports' Designated Bridge Identifiers from comparisons.

NOTE 2—This ability of MSTP Bridges to communicate the full set of MSTP information on shared LANs to which RSTP Bridges are attached avoids the need for the Port Protocol Migration machines to detect RSTP Bridges. Two or more MSTP and one or more RSTP Bridges may be connected to a shared LAN, with full MSTP operation. This includes the possibility of different MSTI Designated Ports (see 13.5.1).

13.6.2 Force Protocol Version

A Force Protocol Version parameter, controlled by management, permits emulation of aspects of the behavior of earlier versions of spanning tree protocol that are not strictly required for interoperability. The value of this parameter applies to all Bridge Ports.

- a) STP BPDUs, rather than MST BPDUs, are transmitted if Force Protocol Version is 0. RST BPDUs omit the MST Configuration Identifier and all MSTI Information.
- b) RST BPDUs, rather than MST BPDUs, are transmitted if Force Protocol Version is 2. RST BPDUs omit the MST Configuration Identifier and all MSTI Information.
- c) All received BPDUs are treated as being from a different MST Region if Force Protocol Version is 0 or 2.

- d) Rapid transitions are disabled if Force Protocol Version is 0. This allows MSTP Bridges to support applications and protocols that can be sensitive to the increased rates of frame duplication and misordering that can arise under some circumstances, as discussed in Annex K of IEEE Std 802.1D-2004.
- e) The MSTP state machines allow full MSTP behavior if Force Protocol Version is 3 or more.

NOTE—Force Protocol Version does not support multiple spanning trees with rapid transitions disabled.

13.7 MST Configuration Identifier

It is essential that all bridges within an MST Region (3.113) agree on the allocation of VIDs to spanning trees. If the allocation differs, frames for some VIDs may be duplicated or not delivered to some LANs at all. MST Bridges check that they are allocating VIDs to the same spanning trees as their neighbors in the same region by transmitting and receiving MST Configuration Identifiers in BPDUs. Each MST Configuration Identifier includes a Configuration Digest that is compact but designed so that two matching identifiers have a very high probability of denoting the same allocation of VIDs to MSTIDs (3.118, 8.4) even if the identifiers are not explicitly managed. Suitable management practices for equipment deployment and for choosing Configuration Names and Revision Levels (see below) can guarantee that the identifiers will differ if the VID to tree allocation differs within a single administrative domain.

Each MST Configuration Identifier contains the following components:

- 1) A Configuration Identifier Format Selector, the value 0 encoded in a fixed field of one octet to indicate the use of the following components as specified in this standard.
- 2) The Configuration Name, a variable length text string encoded within a fixed field of 32 octets, conforming to RFC 2271's definition of SnmpAdminString. If the Configuration Name is less than 32 characters, the text string should be terminated by the NUL character, with the remainder of the 32-octet field filled with NUL characters. Otherwise the text string is encoded with no terminating NUL character.
- 3) The Revision Level, an unsigned integer encoded within a fixed field of 2 octets.
- 4) The Configuration Digest, a 16-octet signature of type HMAC-MD5 (see IETF RFC 2104 (1997)) created from the MST Configuration Table (3.112, 8.9). To calculate the digest, the table is considered to contain 4096 consecutive two octet elements, where each element of the table (with the exception of the first and last) contains an MSTID value encoded as a binary number, with the first octet being most significant. The first element of the table contains the value 0, the second element the MSTID value corresponding to VID 1, the third element the MSTID value corresponding to VID 2, and so on, with the next to last element of the table containing the MSTID value corresponding to VID 4094, and the last element containing the value 0. The key used to generate the signature consists of the 16-octet string specified in Table 13-1.

Table 13-1—Configuration Digest Signature Key

Parameter	Mandatory value
Configuration Digest Signature Key	0x13AC06A62E47FD51F95D2BA243CD0346

NOTE—The formulation of the signature as described above does not imply that a separate VID to MSTID translation table has to be maintained by the implementation; rather that it should be possible for the implementation to derive the logical contents of such a table, and the signature value as specified above, from the other configuration information maintained by the implementation, as described in Clause 12.

The Configuration Digests of some VID to MSTID translations are shown in Table 13-2 to help verify implementations of this specification.

Table 13-2—Sample Configuration Digest Signature Keys

VID to MSTID translation	Configuration Digest
All VIDs map to the CIST, no VID mapped to any MSTI	0xAC36177F50283CD4B83821D8AB26DE62
All VIDs map to MSTID 1	0xE13A80F11ED0856ACD4EE3476941C73B
Every VID maps to the MSTID equal to (VID modulo 32) + 1	0x9D145C267DBE9FB5D893441BE3BA08CE

It is recommended that MST Bridge implementations provide an easily selectable or default configuration comprising a Configuration Name of the Bridge Address as a text string using the Hexadecimal Representation specified in IEEE Std 802, a Revision Level of 0, and a Configuration Digest representing a VID to MSTID translation table containing the value 0 for every element. Such a table represents the mapping of all VIDs to the CIST. Since the Bridge Address is unique to each bridge, no two bridges using this default configuration will be identified as belonging to the same region.

13.8 Spanning Tree Priority Vectors

Priority vectors permit concise specification of each protocol's computation of the active topology, both in terms of the entire network and of the operation of individual bridges in support of the distributed algorithm. MST, RST, and STP bridges use *spanning tree priority vector* information in Configuration Messages (13.13), sent and received from neighboring bridges, to assign Port Roles that determine each port's participation in a fully and simply connected active topology based on one or more spanning trees.

CIST priority vectors comprise the following components:

- a) CIST Root Identifier, the Bridge Identifier of the CIST Root;
- b) CIST External Root Path Cost, the inter-regional cost from the transmitting bridge to the CIST Root;
- c) CIST Regional Root Identifier, the Bridge Identifier of the single bridge in a region whose CIST Root Port connects to a LAN in a different region, or of the CIST Root if that is within the region;
- d) CIST Internal Root Path Cost, the cost to the CIST Regional Root;
- e) CIST Designated Bridge Identifier, the Bridge Identifier for the transmitting bridge for the CIST;
- f) CIST Designated Port Identifier, the Port Identifier for the transmitting port for the CIST;
- g) CIST Receiving Port Identifier (not conveyed in Configuration Messages, used as tie-breaker between otherwise equal priority vectors within a receiving bridge).

The first two components of the CIST priority vector are significant throughout the network. The CIST External Root Path Cost transmitted by a bridge is propagated along each path from the CIST Root, is added to at Bridge Ports that receive the priority vector from a bridge in a different region, and thus accumulates costs at the Root Ports of bridges that are not MST or are CIST Regional Roots and is constant within a region. The CIST Internal Root Path Cost is only significant and defined within a region. The last three components are used as locally significant tie breakers, not propagated within or between regions. The set of all CIST spanning tree priority vectors is thus totally ordered.

Since RSTP is not aware of regions, RSTP specifications also refer to the CIST Root Identifier and CIST External Root Path Cost simply as the Root Bridge Identifier and Root Path Cost, respectively, and omit the CIST Internal Root Path Cost (as does STP). MSTP encodes the CIST Regional Root Identifier in the BPDU

field used by RSTP to convey the Designated Bridge Identifier (14.3.3), so an entire region appears to an RSTP capable bridge as a single bridge. RSTP's CST use of CIST priority vectors can be conveniently specified by the use of the zero for the Internal Root Path Cost and the same values for both the Regional Root Identifier and Designated Bridge Identifier.

NOTE 1—The path to the CIST Root from a bridge with a CIST Root Port within a region always goes to or through the CIST Regional Root.

NOTE 2—STP lacks the fields necessary for MST Bridges to communicate the Designated Bridge Identifier to resolve a potential priority vector tie, and MSTP BPDUs are not sent on a LAN to which an STP bridge is attached.

Each MSTI priority vector comprises the following components for the particular MSTI in a given region:

- h) MSTI Regional Root Identifier, the Bridge Identifier of the MSTI Regional Root;
- i) MSTI Internal Root Path Cost, the path cost to the MSTI Regional Root;
- j) MSTI Designated Bridge Identifier, the Bridge Identifier for the transmitting bridge for this MSTI;
- k) MSTI Designated Port Identifier, the Port Identifier for the transmitting port for this MSTI;
- l) MSTI Receiving Port Identifier (not conveyed in Configuration Messages).

The set of priority vectors for a given MSTI is only defined within a region. Within each region they are totally and uniquely ordered. A CIST Root Identifier, CIST External Root Path Cost, and CIST Regional Root Identifier tuple defines the connection of the region to the external CST and is required to be associated with the source of the MSTI priority vector information when assessing the agreement of information for rapid transitions to forwarding, but plays no part in priority vector calculations.

As each bridge and Bridge Port receives priority vector information from bridges and ports closer to the Root, calculations and comparisons are made to decide which priority vectors to record, and what information to pass on. Decisions about a given port's role are made by comparing the priority vector components that could be transmitted with that received by the port. For all components, a lesser numerical value is better, and earlier components in the above lists are more significant. As each Bridge Port receives information from ports closer to the Root, additions are made to one or more priority vector components to yield a worse priority vector for potential transmission through other ports of the same bridge.

NOTE 3—The consistent use of lower numerical values to indicate better information is deliberate as the Designated Port that is closest to the Root Bridge, i.e., has a numerically lowest path cost component, is selected from among potential alternatives for any given LAN (13.8). Adopting the conventions that lower numerical values indicate better information, that where possible more significant priority components are encoded earlier in the octet sequence of a BPDU (14.3), and that earlier octets in the encoding of individual components are more significant (14.2) allow concatenated octets that compose a priority vector to be compared as if they were a multiple octet encoding of a single number, without regard to the boundaries between the encoded components. To reduce the confusion that naturally arises from having the lesser of two numerical values represent the better of the two, i.e., that chosen all other factors being equal, this clause uses the following consistent terminology. Relative numeric values are described as “least,” “lesser,” “equal,” and “greater,” and their comparisons as “less than,” “equal to,” or “greater than,” while relative Spanning Tree priorities are described as “best,” “better,” “the same,” “different,” and “worse” and their comparisons as “better than,” “the same as,” “different from,” and “worse than.” The operators “<” and “=” represent less than and equal to, respectively. The terms “superior” and “inferior” are used for comparisons that are not simply based on priority but include the fact that a priority vector can replace an earlier vector transmitted by the same Bridge Port. All of these terms are defined for priority vectors in terms of the numeric comparison of components below (13.9, 13.10).

NOTE 4—To ensure that the CIST and each MSTI's view of the boundaries of each region remain in synchronization at all times, each BPDU carries priority vector information for the CIST as well as for MSTIs. Associating the CIST Root Identifier, External Path Cost, and Regional Root Identifier with the priority vector information for each MSTI does not therefore raise a requirement to transmit these components separately. A single bit per MSTI vector, the Agreement flag, satisfies the requirement to indicate that the vector beginning with the MSTI Regional Root Identifier for that specific MSTI has always been associated with the single CIST Root Identifier, etc. transmitted in the BPDU.

To allow the active topology to be managed for each tree through adjusting the relative priority of different bridges and Bridge Ports for selection as the CIST Root, a CIST or MSTI Regional Root, Designated

Bridge, or Designated Port, the priority component of the bridge's Bridge Identifier can be independently chosen for the CIST and for each MSTI. The priority component used by the CIST for its CIST Regional Root Identifier can also be chosen independently of that used for the CIST Root Identifier. Independent configuration of Port Path Cost and Port Priority values for the CIST and for each MSTI can also be used to control selection of the various roles for the CIST and for each MSTI.

13.9 CIST Priority Vector calculations

The *port priority vector* is the priority vector held for the port when the reception of BPDUs and any pending update of information has been completed:

$$\begin{aligned} \text{port priority vector} = & \{ \text{RootID} : \text{ExtRootPathCost} : \\ & \quad \text{RRootID} : \text{IntRootPathCost} : \\ & \quad \text{DesignatedBridgeID} : \text{DesignatedPortID} : \text{RcvPortID} \} \end{aligned}$$

The *message priority vector* is the priority vector conveyed in a received Configuration Message. For a bridge with Bridge Identifier B receiving a Configuration Message on a port P_B from a Designated Port P_D on bridge D claiming a CIST Root Identifier of R_D , a CIST External Root Path Cost of ERC_D , a CIST Regional Root Identifier of RR_D , and a CIST Internal Root Path Cost of IRC_D :

$$\text{message priority vector} = \{ R_D : ERC_D : RR_D : IRC_D : D : P_D : P_B \}$$

If B is not in the same region as D , the Internal Root Path Cost has no meaning to B and is set to 0.

NOTE—If a Configuration Message is received in an RST or STP BPDU, both the Regional Root Identifier and the Designated Bridge Identifier are decoded from the single BPDU field used for the Designated Bridge Parameter (the MST BPDU field in this position encodes the CIST Regional Root Identifier). An STP or RSTP bridge is always treated by MSTP as being in an region of its own, so the Internal Root Path Cost is decoded as zero.

The received CIST message priority vector is the same as B 's port priority vector if:

$$\begin{aligned} (R_D == \text{RootID}) \&\& (ERC_D == \text{ExtRootPathCost}) \&\& (RR_D == \text{RRootID}) \&\& \\ (IRC_D == \text{IntRootPathCost}) \&\& (D == \text{DesignatedBridgeID}) \&\& (P_D == \text{DesignatedPortID}) \end{aligned}$$

and is better if:

$$\begin{aligned} ((R_D < \text{RootID})) \mid\mid \\ ((R_D == \text{RootID}) \&\& (ERC_D < \text{ExtRootPathCost})) \mid\mid \\ ((R_D == \text{RootID}) \&\& (ERC_D == \text{ExtRootPathCost}) \&\& (RR_D < \text{RRootID})) \mid\mid \\ ((R_D == \text{RootID}) \&\& (ERC_D == \text{ExtRootPathCost}) \&\& (RR_D == \text{RRootID}) \\ \quad \&\& (IRC_D < \text{IntRootPathCost})) \mid\mid \\ ((R_D == \text{RootID}) \&\& (ERC_D == \text{ExtRootPathCost}) \&\& (RR_D == \text{RRootID}) \\ \quad \&\& (IRC_D == \text{IntRootPathCost}) \&\& (D < \text{DesignatedBridgeID})) \mid\mid \\ ((R_D == \text{RootID}) \&\& (ERC_D == \text{ExtRootPathCost}) \&\& (RR_D == \text{RRootID}) \\ \quad \&\& (IRC_D == \text{IntRootPathCost}) \&\& (D == \text{DesignatedBridgeID}) \\ \quad \&\& (P_D < \text{DesignatedPortID})) \end{aligned}$$

A received CIST message priority vector is superior to the port priority vector if, and only if, the message priority vector is better than the port priority vector, or the Designated Bridge Identifier Bridge Address and Designated Port Identifier Port Number components are the same; in which case, the message has been transmitted from the same Designated Port as a previously received superior message, i.e., if:

$$\begin{aligned} \{ R_D : ERC_D : RR_D : IRC_D : D : P_D : P_B \} \\ \text{is better than} \end{aligned}$$

```

{RootID : ExtRootPathCost : RRootID : IntRootPathCost :
 DesignatedBridgeID : DesignatedPortID : RcvPortID}
 )|| ((D.BridgeAddress == DesignatedBridgeID.BridgeAddress) &&
 (P_D.PortNumber == DesignatedPortID.PortNumber))

```

If the message priority vector received in a Configuration Message from a Designated Port is superior, it will replace the current port priority vector.

A *root path priority vector* for a port can be calculated from a port priority vector that contains information from a message priority vector, as follows:

If the port priority vector was received from a bridge in a different region (13.27.8), the External Port Path Cost EPC_{PB} is added to the External Root Path Cost component, and the Regional Root Identifier is set to the value of the Bridge Identifier for the receiving bridge. The Internal Root Path Cost component will have been set to zero on reception.

$$\text{root path priority vector} = \{R_D : ERC_D + EPC_{PB} : B : 0 : D : P_D : P_B\}$$

If the port priority vector was received from a bridge in the same region (13.27.8), the Internal Port Path Cost IPC_{PB} is added to the Internal Root Path Cost component.

$$\text{root path priority vector} = \{R_D : ERC_D : RR_D : IRC_D + IPC_{PB} : D : P_D : P_B\}$$

The *bridge priority vector* for a bridge B is the priority vector that would, with the Designated Port Identifier set equal to the transmitting Port Identifier, be used as the message priority vector in Configuration Messages transmitted on bridge B 's Designated Ports if B was selected as the Root Bridge of the CIST.

$$\text{bridge priority vector} = \{B : 0 : B : 0 : B : 0 : 0\}$$

The *root priority vector* for bridge B is the best priority vector of the set of priority vectors comprising:

- a) the bridge priority vector; plus
- b) all root path priority vectors that have a Designated Bridge Identifier D that is not equal to B .

If the bridge priority vector is the best of this set of priority vectors, Bridge B has been selected as the CIST Root.

The *designated priority vector* for a port Q on bridge B is the root priority vector with B 's Bridge Identifier B substituted for the *DesignatedBridgeID* and Q 's Port Identifier Q_B substituted for the *DesignatedPortID* and *RcvPortID* components. If Q is attached to a LAN that has one or more STP bridges attached (as determined by the Port Protocol Migration state machine), B 's Bridge Identifier B is also substituted for the *RRootID* component.

If the designated priority vector is better than the port priority vector, the port will be the Designated Port for that LAN and the current port priority vector will be updated. The message priority vector in Configuration Messages transmitted by a port always comprises the components of the designated priority vector for the port, even if the port is a Root Port.

13.10 MST Priority Vector calculations

The *port priority vector* for a given MSTI is the priority vector held for the port per MSTI when the reception of BPDUs and any pending update of information has been completed:

port priority vector = { RR_{RootID} : $IntRootPathCost$:
 $DesignatedBridgeID$: $DesignatedPortID$: $RcvPortID$ }

The *message priority vector* for a given MSTI is the MSTI priority vector conveyed in a received Configuration Message. For a bridge with Bridge Identifier B receiving a Configuration Message on a Regional Port P_B from a Designated Port P_D on bridge D belonging to the same MST Region (3.113) and claiming an Internal Root Path Cost of IRC_D :

message priority vector = { RR_D : IRC_D : D : P_D : P_B }

An MSTI message priority vector received from a bridge not in the same MST Region is discarded.

An MSTI message priority vector received from a Bridge Port internal to the region is the same as the port priority vector if:

(($RR_D == RR_{RootID}$) && ($IRC_D == IntRootPathCost$) && ($D == DesignatedBridgeID$)
&& ($P_D == DesignatedPortID$))

and is better if:

(($RR_D < RR_{RootID}$) ||
(($RR_D == RR_{RootID}$) && ($IRC_D < IntRootPathCost$) ||
(($RR_D == RR_{RootID}$) && ($IRC_D == IntRootPathCost$) && ($D < DesignatedBridgeID$)) ||
(($RR_D == RR_{RootID}$) && ($IRC_D == IntRootPathCost$) && ($D == DesignatedBridgeID$)
&& ($P_D < DesignatedPortID$)))

An MSTI message priority vector is superior to the port priority vector if, and only if, the message priority vector is better than the port priority vector, or the Designated Bridge Identifier Bridge Address and Designated Port Identifier Port Number components are the same; in which case, the message has been transmitted from the same Designated Port as a previously received superior message, i.e., if:

{ RR_D : IRC_D : D : P_D : P_B }
is better than
{ RR_{RootID} : $IntRootPathCost$: $DesignatedBridgeID$: $DesignatedPortID$: $RcvPortID$ }
) || (($D.BridgeAddress == DesignatedBridgeID.BridgeAddress$) &&
($P_D.PortNumber == DesignatedPortID.PortNumber$))

If the message priority vector received in a Configuration Message from a Designated Port for the MSTI is superior, it will replace the current port priority vector.

NOTE 1—The **agree** flag (13.25.4) for the port and this MSTI will be cleared if the CIST Root Identifier, CIST External Root Path Cost, and CIST Regional Root Identifier in the received BPDU are not better than or the same as those for the CIST designated priority vector for the port following processing of the received BPDU.

A *root path priority vector* for a given MSTI can be calculated for a port that has received a port priority vector from a bridge in the same region by adding the Internal Port Path Cost IPC_{PB} to the Internal Root Path Cost component.

root path priority vector = { RR_D : $IRC_D + IPC_{PB}$: D : P_D : P_B }

NOTE 2—Internal Port Path Costs are independently manageable for each MSTI, as are the priority components of the Bridge and Port Identifiers. The ability to independently manage the topology of each MSTI without transmitting individual Port Path Costs is a key reason for retaining the use of a Distance Vector protocol for constructing MSTIs. A simple Link State Protocol requires transmission (or *a priori* sharing) of all Port Costs for all links.

The *bridge priority vector* for a bridge B for a given MSTI is the priority vector that would, with the Designated Port Identifier set equal to the transmitting Port Identifier, be used as the message priority vector in Configuration Messages transmitted on bridge B 's Designated Ports if B was selected as the Root Bridge of a given tree.

$$\text{bridge priority vector} = \{B : 0 : B : 0\}$$

The *root priority vector* for bridge B is the best priority vector of the set of priority vectors comprising the bridge priority vector plus all root path priority vectors whose Designated Bridge Identifier D is not equal to B . If the bridge priority vector is the best of this set of priority vectors, Bridge B has been selected as the Root of the tree.

The *designated priority vector* for a port Q on bridge B is the root priority vector with B 's Bridge Identifier B substituted for the *DesignatedBridgeID* and Q 's Port Identifier Q_B substituted for the *DesignatedPortID* and *RcvPortID* components.

If the designated priority vector is better than the port priority vector, the port will be the Designated Port for the attached LAN and the current port priority vector will be updated. The message priority vector in MSTP BPDUs transmitted by a port always comprises the components of the designated priority vector of the port, even if the port is a Root Port.

Figure 13-8 shows the priority vectors and the active topology calculated for an MSTI in a region of the example network of Figure 13-6.

13.11 Port Role assignments

Each bridge assigns CIST Port Roles (when new information becomes available as specified in this clause, 13.11) before assigning MSTI Port Roles. The calculations specified in 13.9 are used to assign a role to each Bridge Port that is enabled as follows:

- a) If the bridge is not the CIST Root, the source of the root priority vector is the Root Port.
- b) Each port whose port priority vector is the designated priority vector is a Designated Port.
- c) Each port, other than the Root Port, with a port priority vector received from another bridge is an Alternate Port.
- d) Each port with a port priority vector received from another port on this bridge is a Backup Port.

If the port is not enabled, i.e. its MAC_Operational status is FALSE or its Administrative Bridge Port state is Disabled (8.4), it is assigned the Disabled Port role for the CIST and all MSTIs, to identify it as having no part in the operation of any of the spanning trees or the active topology of the network.

If the bridge is an MST Bridge, the calculations specified in 13.10 are used to assign a role to each enabled Bridge Port for each MSTI as follows:

- e) If the port is the CIST Root Port and the CIST port priority vector was received from a bridge in another MST, the port is the Master Port.
- f) If the bridge is not the MSTI Regional Root, the port that is the source of the MSTI root priority vector is the Root Port.
- g) Each port whose port priority vector is the designated priority vector derived from the root priority vector is a Designated Port.
- h) Each port, other than the Master Port or the Root Port, with a port priority vector received from another bridge or a CIST port priority vector from a bridge in another region, is an Alternate Port.
- i) Each port that has a port priority vector that has been received from another port on this bridge is a Backup Port.

13.12 Stable connectivity

This clause provides an analysis to show that RSTP and MSTP meet the goal of providing full and simple connectivity for frames assigned to any given VID in a stable network, i.e., where the physical topology has remained constant for long enough that the spanning tree information communicated and processed by bridges is not changing.

NOTE 1—The FDB can be configured to prevent connectivity, in particular this analysis assumes that every Bridge Port is a member of every VID's Member Set (8.8.10). Spanning tree protocol controls can also be used to prevent new connectivity (to allow for upgrades), or to disallow certain topologies (restricting the location of the CIST Root, for example). This analysis assumes that those controls are not being used, that all the bridges are using conformant protocol implementations and that the LANs are providing omnidirectional connectivity.

Every LAN provides connectivity for all frames between all attached Bridge Ports. Every bridge provides connectivity between and only between its CIST Root and Designated Ports for frames assigned to the CIST, and between the Root, Designated, and Master Ports for a given MSTI for frames assigned to that MSTI. Any given bridge does not assign frames to more than one tree and has one Root Port per tree, unless it is the Root of that tree.

Every LAN has one and only one CIST Designated Port, and every bridge apart from the CIST Root has one and only one CIST Root Port. The CIST spanning tree priority vector of the Designated Port attached to the LAN that is connected to a bridge's Root Port is better than of any Designated Port of that bridge. The CIST thus connects all bridges and LANs (is “spanning”) and loop-free (is a “tree”).

Each MST Region is bounded by CST Root and Alternate Ports. At the CST Root Ports connectivity for frames assigned to MSTIs within the connected regions is the same as that for the CIST. Every region apart from that containing the CIST Root has a single CST Root Port, identified as the Master Port for each MSTI. The CIST spanning tree priority vector of the LAN attached to the region's CST Root Port is better than that of any CST Designated Port of a bridge in the region attached to a LAN also attached to the CST Root Port of another region. The CST thus provides loop-free connectivity between all regions.

NOTE—The term “Common Spanning Tree (CST)” refers to the CIST connectivity between regions, and the term “Internal Spanning Tree (IST)” to the CIST connectivity within each region. An RSTP bridge and the LANs for which it is the Designated Bridge are conveniently considered as forming an MST region of limited extent.

Within each region each frame is consistently assigned to the CIST, or an MSTI, and each of these spanning trees provides full loop-free connectivity to each of the bridges within the region, just as the CIST does for the network as a whole, including connectivity between the CST Root Port (Master Port) and the CST Designated Ports. Since each bridge or LAN is in one and only one region, and it has already been shown that loop-free connectivity is provided between regions, loop-free connectivity is thus provided between all the bridges and LANs in the network.

Figure 13-9 illustrates the above connectivity with the simple example of Region 1 from the example network of Figure 13-6 and Figure 13-8. Bridge 0.42 has been selected as the CIST Root and Regional Root, bridge 0.57 as the Regional Root for MSTI 1, and bridge 2.83 for MSTI 2 by management of the per MSTI Bridge Identifier priority component. The potential loop through the three bridges in the region is blocked at different Bridge Ports for the CIST, and each MSTI, but the connectivity across the region and from each LAN and bridge in the region through the boundaries of the region is the same in all cases.

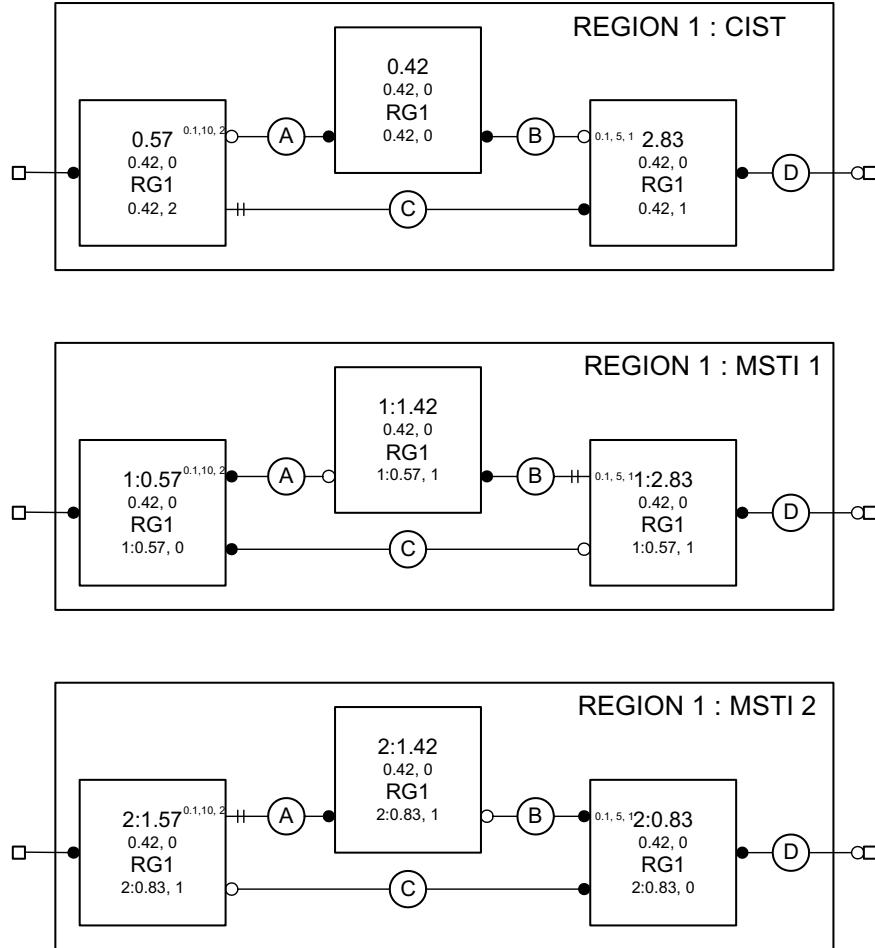


Figure 13-9—CIST and MSTI active topologies in Region 1 of the example network

13.13 Communicating Spanning Tree information

A Spanning Tree Protocol Entity transmits and receives group addressed BPDUs (Clause 14, 8.13.5) through each of its Bridge Ports to communicate with the Spanning Tree Protocol Entities of the other bridges attached to the same LAN. The group address used is one of a small number of addresses that identify frames that are not directly forwarded by bridges (8.6.3), but the information in the BPDU can be used by a bridge in calculating its own BPDUs to transmit and can stimulate that transmission.

BPDUs are used to convey the following:

- Configuration Messages
- Topology Change Notification (TCN) Messages
- MST Configuration Identifiers

Designated Ports also transmit BPDUs at intervals to guard against loss and to assist in the detection of failed components (LANs, bridges, or Bridge Ports), so all messages are designed to be idempotent.

A Configuration Message for the CIST can be encoded in an STP Configuration BPDU, an RST BPDU, or an MST BPDU (14.5, 14.6). A TCN Message for the CIST can be encoded in an STP Topology Change Notification BPDU (14.5), or an RST or MST BPDU with the TC flag set. Configuration and TCN

Messages for the CIST and for all MSTIs in an MST Region are encoded in a single MST, as is the MST Configuration Identifier. No more than 64 MSTI Configuration Messages shall be encoded in an MST BPDU, and no more than 64 MSTIs shall be supported by an MST Bridge.

Configuration and Topology Change Notification BPDUs are distinguished from each other and from RST and MST BPDUs by their BPDU Type (Clause 14). RST and MST BPDUs share the same BPDU Type and are distinguished by their version identifiers.

Bridges implementing STP (Clause 8 of IEEE Std 802.1D, 1998 Edition) transmit and decode Configuration and Topology Change Notification BPDUs, and ignore RST and MST BPDUs on receipt. This ensures that connection of a Bridge Port of such a bridge to a LAN that is also attached to a bridge implementing RSTP or MSTP is detected, as transmission of RSTP or MSTP BPDUs does not suppress regular transmissions by the STP bridge. This functionality is provided by the Port Protocol Migration state machine for RSTP (13.30). The Port Protocol Migration state machines select the BPDU types used to encode Spanning Tree messages so that all bridges attached to the same LAN participate in a spanning tree protocol, while maximizing the available functionality. If one or more attached bridges only implement STP, only Configuration and Topology Change Notification BPDUs will be used and the functionality provided by the protocol will be constrained.

13.14 Changing Spanning Tree information

Addition, removal, failure, or management of the parameters of bridges and LAN connectivity can change spanning tree information and require Port Role changes in all or part of the network (for the CIST) or all or part of an MST Region (for an MSTI). A CIST or MSTI configuration message received in a BPDU is considered superior to, and will replace, that recorded in the reception Port's port priority vector if its message priority vector is better, or if it was transmitted by the same Designated Bridge and Designated Port and the message priority vector, timer, or hop count information differ from those recorded.

RSTP and MSTP propagate new information rapidly from bridge to bridge, superseding prior information and stimulating further transmissions until it reaches either Designated Ports that have already received the new information through redundant paths in the network or the leaves of the Spanning Tree, as defined by the new configuration. Configuration Message transmissions will then once more occur at regular intervals from ports selected as Designated Ports.

To ensure that old information does not endlessly circulate through redundant paths in the network, preventing the effective propagation of the new information, MSTP associates a hop count with the information for each spanning tree. The hop count is assigned by the CIST Regional Root or the MSTI Regional Root and decremented by each reception Port. Received information is discarded and the port made a Designated Port if the hop count reaches zero.

RSTP and MSTP's CST processing do not use an explicit hop count (for reasons of STP compatibility), but detect circulating aged information by treating the BPDU Message Age parameter as an incrementing hop count with Max Age as its maximum value. MSTP increments Message Age for information received at the boundary of an MST Region, discarding the information if necessary.

If a Bridge Port's MAC_Operational parameter becomes FALSE, the port becomes a Disabled Port and received information is discarded. Spanning tree information for the tree can be recomputed, the bridge's Port Roles changed, and new spanning tree information transmitted if necessary. Not all component failure conditions can be detected in this way, so each Designated Port transmits BPDUs at regular intervals and a reception Port will discard information and become a Designated Port if two transmissions are missed.

NOTE—Use of a separate hop count and message loss detection timer provides superior reconfiguration performance compared with the original use of Message Age and Max Age by STP. Connectivity loss detection is not compromised by the need to allow for the overall diameter of the network, nor does the time allowed extend the number of hops

permitted to aged recirculating information. Management calculation of the necessary parameters for custom topologies is also facilitated, as no allowance needs to be made for relative timer jitter and accuracy in different bridges.

13.15 Changing Port States with RSTP or MSTP

The Port State for the CIST and each MSTI for each Bridge Port is controlled by state machines whose goal is to maximize connectivity without introducing temporary loops in each of these active topologies. Root Ports, Master Ports, and Designated Ports are transitioned to the Forwarding Port State, and Alternate Ports and Backup Ports to the Discarding Port State, as rapidly as possible. Transitions to the Discarding Port State can be simply effected without the risk of data loops. This clause (13.15) describes the conditions that RSTP and MSTP use to transition a Port State for a given spanning tree to Forwarding.

Starting with the assumption that any connected fragment of a network is composed of bridges, Bridge Ports, and connected LANs that form a subtree of a spanning tree, ports with Root Port, Master Port, or Designated Port roles are transitioned using conditions that ensure that the newly enlarged fragment continues to form either a subtree or the whole of the spanning tree. Since the conditions are used every time a fragment is enlarged, it is possible to trace the growth of a fragment from a single bridge—a consistent, if small, subtree of a spanning tree—to any sized fragment, thus justifying the initial assumption.

Port States in two subtrees, each bounded by ports that are not forwarding or are attached to LANs not attached to any other bridge, can be made consistent by waiting for any changes in the priority vector information used to assign Port Roles to reach all bridges in the network, thus ensuring that the subtrees are not, and are not about to be, joined by other Forwarding Ports. However, it can be shown that a newly selected Root Port can forward frames as soon as prior recent root ports on the same bridge cease to do so, without further communication from other bridges. Rapid transitions of Designated Ports and Master Ports do require an explicit Agreement from the bridges in the subtrees to be connected. The Agreement mechanism is described, together with a Proposal mechanism that forces satisfaction of the conditions if they have not already been met by blocking Designated Ports connecting lower subtrees that are not yet in agreement. The same Agreement mechanism is then used to transition the newly blocked ports back to forwarding, advancing any temporary cut in the active topology toward the edge of the network.

13.15.1 Subtree connectivity and priority vectors

Any given bridge B , the LANs connected through its Forwarding Designated Ports, the further bridges connected to those LANs through their Root Ports, the LANs connected to their Forwarding Designated Ports, and so on, recursively, constitute a subtree S_B . Any LAN L that is part of S_B will be connected to B through a Forwarding Designated Port P_{CL} on a bridge C also in S_B . L cannot be directly connected to any port P_B on bridge B unless B and C are one and the same, since the message priority vector for P_B is better than that of any port of any other bridge in S_B , and prior to Forwarding P_{CL} will have advertised its spanning port priority vector for long enough for it to receive any better message priority vector (within the design probabilities of protocol failure due to repeated BPDU loss) or will have engaged in an explicit confirmed exchange (see below) with all other Bridge Ports attached to that LAN.

13.15.2 Root Port transition to Forwarding

It follows from the above that B 's Root Port can be transitioned to Forwarding immediately whether it is attached to a LAN in S_B or in the rest of the network, provided that all prior recent Root Ports on B (that might be similarly arbitrarily attached) have been transitioned to Discarding and the Root Port was not a Backup Port recently (B and C the same as above).

13.15.3 Designated Port transition to Forwarding

On any given bridge A , the Designated Port P_{AM} connected to a LAN M can be transitioned to Forwarding provided that the message priority advertised by the Designated Port P_{CL} on any LAN L in any subtree S_{M1}, S_{M2} , etc. connected to M is worse than that advertised by P_{AM} ; that any bridge D attached to L has agreed that P_{CL} is the Designated Port; and that only the Root Port and Designated Ports on D are Forwarding. A sufficient condition for P_{AM} to transition to Forwarding is that M is a point-to-point link attached to the Root Port P_{BM} of a bridge B , that the port priority of P_{BM} is the same as or worse than that of P_{AM} , and any port P_{BN} on B is Discarding or similarly attached to a bridge C . P_{BM} signals this condition to P_{AM} by setting the Agreement flag in a Configuration Message carrying P_{BM} 's designated priority and Port Role.

NOTE 1—RSTP and MSTP use adminPointToPointMAC and operPointToPointMAC (6.6.3) to allow the point-to-point status of LANs to be managed and used by the Port Role Transition state machines for Designated Ports. A newly selected Root Port can be transitioned to Forwarding rapidly, even if attached to a shared media LAN.

Figure 13-10 illustrates the generation of an Agreement at a bridge's Root Port from an Agreement received or a Port State of Discarding at each of its Designated Ports, and a Port State of Discarding at each of its Alternate and Backup Ports. A bridge receiving a Proposal transitions any Designated Port not already synchronized to Discarding so it can send the Agreement, and that port solicits an Agreement by sending a Proposal in its turn.

NOTE 2—Agreements can be generated without prior receipt of a Proposal as soon as the necessary conditions are met. Subsequent receipt of a Proposal serves to elicit a further Agreement. If all other ports have already been synchronized (allSynced in Figure 13-10) and the Proposal's priority vector does not convey worse information, synchronization is maintained and there is no need to transition Designated Ports to Discarding once more, or to transmit further Proposals.

13.15.4 Master Port transition to Forwarding

While the connectivity of the CIST from the CIST Regional Root through an MST Region to the rest of the CST comprises a subtree rooted in the CIST Regional Root, the connectivity of the MSTI from the Master Port includes both a subtree below the CIST Regional Root and a subtree rooted in the MSTI Regional Root and connected to the CIST Regional Root by an MSTI Root Port. In the example network of Figure 13-6, this latter subtree continues CST connectivity, from the Master Port on Bridge 86 through to LAN N, for frames allocated to the MSTI within Region 2 (see Figure 13-11). In general either MSTI subtree could be providing CST connectivity through a prior Master Port: the connectivity of both subtrees has to agree with the new CIST Regional Root, before a new Master Port transitions to Forwarding.

NOTE 1—The physical layout shown in the two halves of Figure 13-11 differs in order to reflect the different priorities and logical topologies for the two spanning tree instances. The layout convention is that Designated Ports are shown as horizontal lines, Root Ports as vertical lines, and Alternate Ports as diagonal lines.

Figure 13-12 illustrates the extension of the Agreement mechanism to signal from Designated Ports to Root Ports as well as vice versa. To ensure that an MSTI does not connect alternate Master Ports, an Agreement is only recognized at an MSTI Port when the associated CIST Regional Root information matches that selected by the reception Port. Proposals, eliciting Agreements, necessarily flow from Designated Ports to Root Ports with the propagation of spanning tree information, so a new CIST Regional Root cannot transmit a Proposal directly on its MSTI Root Ports. However, updating a CIST Designated Port's port priority vector with a new Regional Root Identifier forces the port to discard frames for all MSTIs, thus initiating the Proposal from the first bridge nearer the MSTI Regional Root that learns of the new Regional Root.

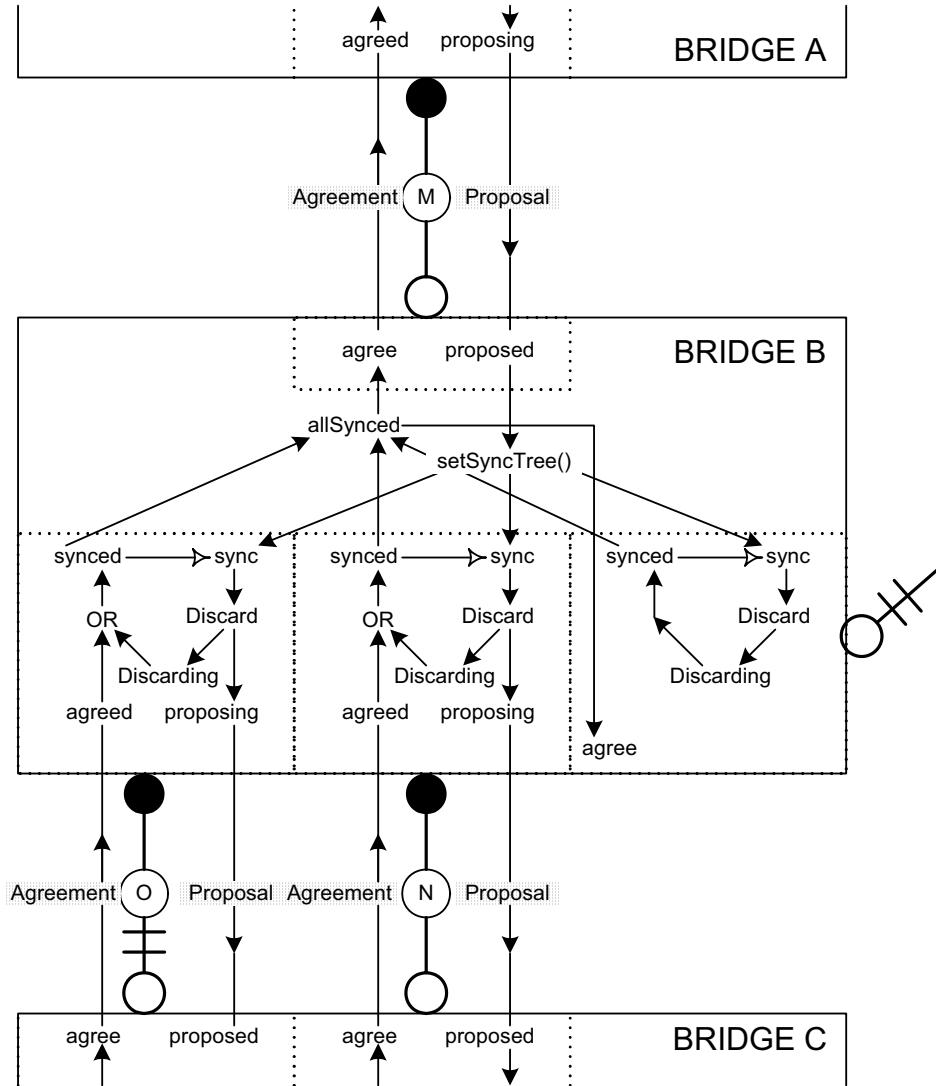


Figure 13-10—Agreements and Proposals

When an Agreement A_{MR} is sent by a Root Port P_{MR} on a Regional Root M , it attests that the CIST Root Identifier and External Root Path Cost components of the message priority advertised on all LANs connected to the CIST by P_{MR} through M are the same as or worse than those accompanying A_{MR} . The connectivity provided by each MSTI can be independent of that provided by the CIST within the MST Region and can therefore connect P_{MR} and one or more CIST Root Ports external to but attached at the boundary of the region even as CIST connectivity within the region is interrupted in order to satisfy the conditions for generating A_{MR} . The Agreement cannot therefore be generated unless all MSTI subtrees as well as the CIST subtree internal to the region are in Agreement. To ensure that an MSTI does not connect to a CIST subtree external to the region that does not meet the constraints on the CST priority vector components, an Agreement received at an MSTI Designated Port from a Bridge Port not internal to the region is only recognized if the CIST Root Identifier and External Root Path Cost of the CIST root priority vector selected by the transmitting Bridge Port are equal to or worse than those selected by the receiver. Updating of a CIST Designated Port's port priority vector with a worse CIST Root Identifier and External Root Path Cost forces the port to discard frames for all MSTIs, thus initiating a Proposal that will elicit agreement.

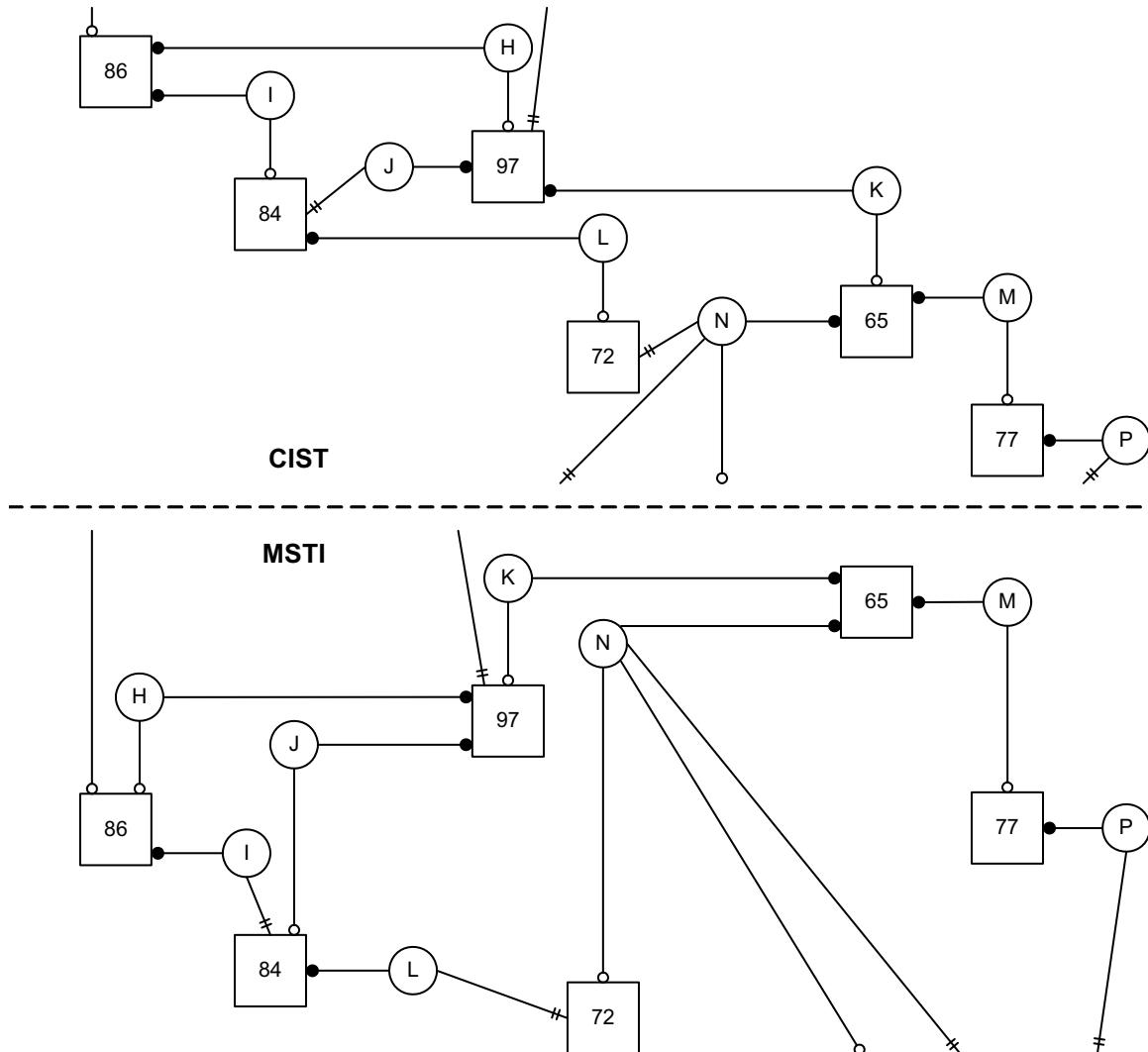


Figure 13-11—CIST and MSTI Active Topologies in Region 2 of Figure 13-6—

NOTE 2—MSTI Designated Ports are prompted to discard frames, as required above, as follows. The CIST Port Information state machine sets sync for all MSTIs on a transition into the UPDATE state if updating the port priority with the designated priority changes the Regional Root Identifier or replaces the CIST Root Identifier or External Path Cost with a worse tuple. The MSTI’s Port Role Transition machine acts on the sync, instructing the port to discard frames, and setting synced and cancelling sync when the port is discarding or an agreement is received.

NOTE 3—A “cut” in an MSTI can be transferred to the CST, either at a Designated Port attached to the same LAN as an STP bridge or at the Root Port of a bridge in an adjacent region. If the CST priority components have already been synced, as is likely if the original cut was caused by changes in physical topology within the region, the cut will terminate there. Otherwise the transferred cut precedes a cut in the CIST, and the synced port may terminate the latter. In that way, cuts in the CST will proceed through an MST Region by the quickest tree that will carry them.

NOTE 4—In the important topology where the CIST Root Bridge is within an MST Region, cuts are not transferred from the region’s IST to any MSTI. CIST cuts propagating in the region will not disrupt MSTI connectivity.

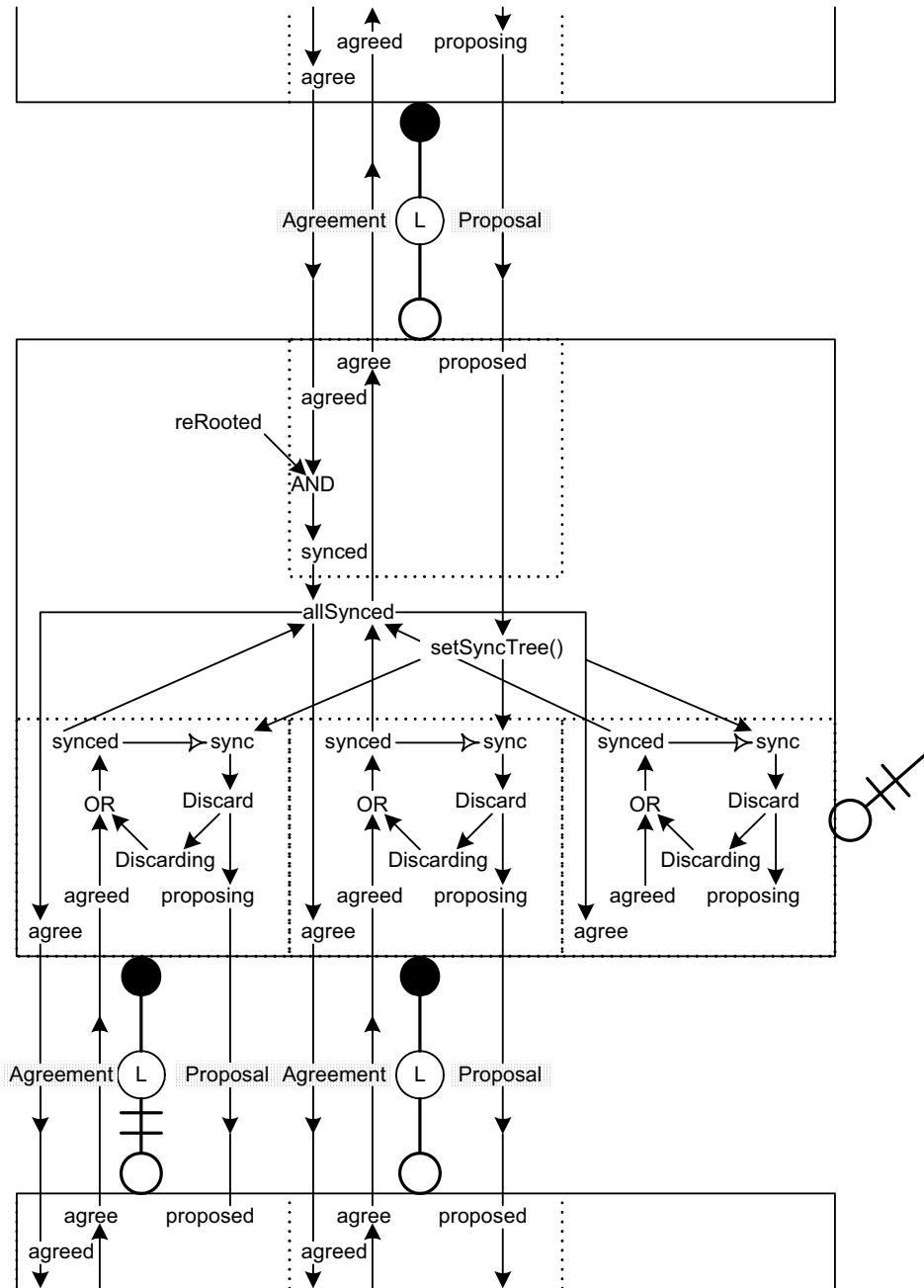


Figure 13-12—Enhanced Agreements

13.16 Managing spanning tree topologies

The active topology of the CIST, and the topologies that can result after the failure or addition of network components, may be managed by assigning values to some or all of the following:

- The Bridge Priority component of the CIST Bridge Identifier for one or more bridges.
- The External Port Path Cost (also referred to as the Port Path Cost for RSTP) for some Bridge Ports.
- Components of the MST Configuration Identifier for bridges with the same Configuration Digest.
- The CIST Internal Port Path Cost Port (for MSTP) for some Bridge Ports.

- The Port Priority component of the Port Identifier for some Bridge Ports.

Within an MST Region, the active topology of each MSTI may be managed by assigning values to some or all of the following:

- The Bridge Priority component of the MSTI Regional Root Identifier.
- The Internal Port Path Cost for the MSTI for some Bridge Ports,
- The Port Priority component of the MSTI's Port Identifier for some Bridge Ports.

In general topology management objectives can be met by modifying only a few parameter values in a few bridges in the network. Table 13-3 specifies default values and ranges for Bridge Priorities and Port Priorities. If these parameters can be updated by management, the bridge shall have the capability to use the full range of values with the granularity specified.

Table 13-3—Bridge and Port Priority values

Parameter	Recommended or default value	Range
Bridge Priority	32 768	0–61 440 in steps of 4096
Port Priority	128	0–240 in steps of 16

NOTE 1—The stated ranges and granularities for Bridge Priority and Port Priority differ from those in IEEE Std 802.1D, 1998 Edition and earlier revisions of that standard. Expressing these values in steps of 4096 and 16 allows consistent management of old and new implementations of this standard; the steps chosen ensure that bits that have been reassigned are not modified, but priority values can be directly compared.

Table 13-4 recommends defaults and ranges for Port Path Cost and Internal Port Path Cost values, chosen according to the speed of the attached LAN, to minimize the administrative effort required to provide reasonable active topologies. If these values can be set by management, the bridge shall be able to use the full range of values in the parameter ranges specified, with a granularity of 1.

Table 13-4—Port Path Cost values

Link Speed	Recommended value	Recommended range	Range
<=100 Kb/s	200 000 000	20 000 000–200 000 000	1–200 000 000
1 Mb/s	20 000 000	2 000 000–200 000 000	1–200 000 000
10 Mb/s	2 000 000	200 000–20 000 000	1–200 000 000
100 Mb/s	200 000	20 000–2 000 000	1–200 000 000
1 Gb/s	20 000	2 000–200 000	1–200 000 000
10 Gb/s	2 000	200–20 000	1–200 000 000
100 Gb/s	200	20–2 000	1–200 000 000
1 Tb/s	20	2–200	1–200 000 000
10 Tb/s	2	1–20	1–200 000 000

When two or more links are aggregated (see IEEE Std 802.1AX), Port Path Cost and Internal Port Path Cost values can be modified to reflect the actual throughput. However, as the primary purpose of Path Cost is to select active topologies, it can be inappropriate to track throughput too closely, as the resultant active topology could fluctuate or differ from that intended by the network administrator. For example, if the network administrator had chosen aggregated links for resilience (rather than for increased data rate), it would be inappropriate to change topology as a result of one of the links in an aggregation failing. Similarly, with links that can autonegotiate their data rate, reflecting such changes of data rate in changes to Path Cost

is not necessarily appropriate. As a default behavior, dynamic changes of data rate should not automatically cause changes in Port Path Cost.

NOTE 2—BPDUs are capable of carrying 32 bits of Root Path Cost information, though IEEE Std 802.1D, 1998 Edition, and its earlier revisions limited the range of the Port Path Cost parameter to a 16-bit unsigned integer value. Table 13-4 uses the full 32-bit range to extend the range of supported link speeds. Additional recommended values can be calculated as $20\ 000\ 000\ 000 / (\text{Link Speed in Kb/s})$. Limiting the range of the Path Cost parameter to 1–200 000 000 ensures that the accumulated Path Cost cannot exceed 32 bits over a concatenation of 20 hops. Where bridges using the IEEE Std 802.1D, 1998 Edition, recommendations and others using Table 13-4 are mixed in the same Bridged Local Area Network, explicit configuration is likely to be necessary to obtain reasonable CST topologies.

13.17 Updating learned station location information

In normal stable operation, learned station location information held in the Filtering Database need only change as a consequence of the physical relocation of stations. It is therefore desirable to employ a long aging time for Dynamic Filtering Entries (8.8.3), especially as many end stations transmit frames following power-up causing the information to be relearned.

However, when the active topology reconfigures, stations can appear to move from the point of view of any given bridge even if that bridge's Port States have not changed. If a Bridge Port is no longer part of an active topology, stations are no longer reachable through that port, and its Dynamic Filtering Entries are removed from that bridge's Filtering Database. Conversely, stations formerly reachable through other ports might be reachable through a newly active port. Dynamic Filtering Entries for the other ports are removed, and RSTP and MSTP transmit Topology Change Notification Messages both through the newly active Port and through the other active Ports on that Bridge. TCNs signal additional connectivity, not just changes in connectivity, as relearning a station's location is only possible if it can be reached, and if that is possible when a port is removed from the active topology another port will be added. A TCN is sent when a Bridge Port joins the active topology, and not before, so that bridges can relearn removed station location information and minimize unnecessary flooding of frames. A bridge that receives a TCN on an active port removes Dynamic Filtering Entries for their other active ports and propagates the TCN through those ports.

NOTE 1—STP allowed for the presence of LAN repeaters that could partition a shared media LAN, thus causing stations to appear to move when the partition was repaired later—with the only Bridge Port changing Port Role or Port State transitioning to Discarding at that time. This scenario does not occur with current technology, and its future likelihood does not justify the use of TCNs to signal connectivity loss. Bridge Ports that participate in the MAC status propagation protocol should be capable of originating TCNs when that protocol signals additional connectivity.

The Topology Change state machine (13.37) avoids removing learned information when ports temporarily revert to Discarding to suppress loops. It treats a port as joining the active topology when it becomes forwarding, and no longer active when it becomes an Alternate, Backup, or Disabled Port and stops forwarding and learning. TCNs are not generated following Edge Port (`operEdge`, 13.25.30) Port State changes, as these do not affect connectivity or station location information in the rest of the network, nor are Dynamic Filtering Entries for Edge Ports removed when TCNs are received.

Dynamic Filtering Entries for MAC addresses previously learned on a Root Port may be modified to move those addresses to an Alternate Port that becomes the new Root Port and a TCN sent only through the new Root Port (and not through other active ports), reducing the need to flood frames. This optimization is possible because a retiring Root Port that becomes Discarding temporarily partitions the active topology into two subtrees, one including all bridges and LANs hitherto reachable through the retiring Root Port, and the other including all the others. If the new Root Port simply provides a new path to the first of these subtrees, its stations will not appear to move from the point of view of bridges in the other subtree. Alternatively if a tree reconfiguration is more complex one or more newly Designated Port will become active and will transmit the necessary TCNs.

NOTE 2—The rules described require removal of potentially invalid learned information for a minimum set of ports on each bridge. A bridge implementation can flush information from more ports than strictly necessary, removing (for example) all Dynamic Filtering Entries rather than just those for the specified ports. This does not result in incorrect operation, but will result in more flooding of frames with unknown destination addresses.

Changes in the active topology of any given MSTI do not change Dynamic Filtering Entries for the CIST or any other MSTI, unless the underlying changes in the physical topology that gave rise to the reconfiguration also cause those trees to reconfigure. Changes to the CST, i.e., the connectivity provided between regions, can cause end station location changes for all trees. Changes to an IST can cause CST end station location changes but do not affect MSTIs in that region unless those trees also reconfigure.

On receipt of a CIST TCN Message from a Bridge Port not internal to the region, or on a change in Port Role for a Bridge Port at the region boundary, TCN Messages are transmitted through each of the other ports of the receiving bridge for each MSTI and the Dynamic Filtering Entries for those ports are removed.

NOTE 3—The port receiving the CIST TCN Message can be a Master Port, a Designated Port attached to the same LAN as an STP bridge, or a Designated Port attached to the same LAN as the Root Ports of bridges in other regions.

TCN Messages for the CIST are always encoded in the same way, irrespective of whether they are perceived to have originated from topology changes internal to the region or outside it. This allows RSTP Bridges whose Root Ports attach to a LAN within an MST Region to receive these TCN Messages correctly.

13.18 Managing reconfiguration

The priority component of the Bridge Identifier can be managed for the CIST, and independently for each MSTI, to allow preferential selection of the first choice Root Bridge and of alternate Roots that will be used if the better choices have failed or lack connectivity. The Port Path Cost of each Bridge Port can also be managed to allow preferential selection of the path from each bridge to the Root, with the priority component of each bridge's Bridge Identifier providing a manageable tie-breaker between equal cost paths.

In the event of LAN or bridge failure, fastest reconfiguration (and thus highest service availability) can usually be provided by a bridge that can substitute a prior Alternate Port for a Root Port that now lacks or provides inferior connectivity to the Root. Figure 13-2 illustrates a simple topology where most failures can be handled in this way. Configuration to place a backup Root Bridge adjacent to the primary (bridges 222 and 111 in the neighbor, respectively) enables Alternate Port to Root Port failover in the other bridges. If the original Root Port and its replacement Alternate Port have the same Root Port Path Cost, bridges further from the Root will not see a topology change. Figure 13-5 illustrates a ring topology, where simple failover copes with few failures. However locating a backup Root Bridge adjacent to the preferred Root remains effective in reducing the effect of any change.

RSTP and MSTP are distance vector protocols: following the failure of a Root Bridge (or selection of a costlier path to the Root) spanning tree priority vectors conveying the prior Root can circulate in the network until that information ages out (i.e. until Message Age exceeds Max Age, or the remainingHops parameter is decremented to zero). To accommodate a wide range of networks, the default setting of these parameters permit 20 hops. However, if the preferred and alternate Roots are located at the center of the network, many networks can be configured with minimal values (6 hops, Table 13-5) so each message can only be received by any node at most twice. In larger networks, for example ring backbones with other bridges redundantly attached to adjacent bridges in the ring, the unwanted effects of recirculating information can be prevented by configuring the restrictedRole parameter (13.25.49). This parameter prevents information from being propagated through ports that should never provide connectivity toward the Root. In Figure 13-3, for example, restrictedRole could be configured for any or all of the Designated Ports.

Other potentially disruptive effects of reconfiguration can be prevented or mitigated by setting the restrictedTcn parameter (13.25.50) for a port.

The configuration of a subtree of a spanning tree defined by the connectivity provided by a given bridge B (say) can be made independent of the rest of the tree by configuring B 's Root Port as a Layer 2 Gateway Port (L2GP). An L2GP Port behaves as if it is in continual receipt of a fixed, manageable, spanning tree priority vector. Provided that this fixed priority vector is worse than the actual priority vector transmitted by the Designated Port for the LAN attached to B 's Root Port, there is no potential for a loop. Each port in the subtree will transmit a worse priority vector. If a received message priority vector is worse than the fixed value the disputed flag is set, causing the L2GP port to transition to Discarding. Thus stability of the spanning tree priority vector information in the subtree is maintained, at the possible expense of the connectivity between the subtree and the rest of the tree. Alternatively, if another bridge within B 's subtree becomes the source of better priority information, causing B 's L2GP port to assume a Designated Port Role, a dispute will exist between the newly propagated spanning tree information and the fixed information for the L2GP port, and will also cause that port to become Discarding.

The L2GP capability is optional, and is primarily intended to be used for service access protection (25.9) when a single customer's Bridged Local Area Network is connected to a Provider Backbone Bridged Network (PBBN) or a Provider Bridged Network (PBN). Two or more Bridge Ports on one or more of the bridges within the customer's network, are connected to a single service instance supported by the provider's network, and are configured as L2GP ports. The L2GP ports' fixed priority vectors are each defined by a different, configured pseudoRootId (13.25.37). Only the L2GP port with the best pseudoRootId can provide connectivity to the provider's network. Spanning tree information from that port will be disseminated throughout the customer's network and will be disputed by the other L2GP port, causing the latter to become Discarding. If all the customer network's ports that are bridged to other networks (such as PBNs or PBBNs) are configured as L2GP ports, those ports neither have to transmit or receive BPDUs to prevent loops—though connectivity through providers' networks to other networks will not be protected. Enabling BPDU reception for each L2GP port prevents loops that might otherwise be caused by attaching another of the customer's network's ports to the provider network.

13.19 Partial and disputed connectivity

It is possible for the connectivity between Bridge Ports attached to the same LAN to fail, in system or media access method dependent ways, so that BPDUs and data frames are transmitted in one direction only. As a result it is possible for more than one port attached to the same LAN to believe itself to be the Designated Port. To ensure that the active topology remains loop-free, a Designated Port will recognise that a dispute is in progress and stop learning from or forwarding frames, if it receives a BPDU with a worse message priority and the Learning flag set from another port that claims to be Designated.

If two (Designated) ports attached to the same shared media LAN cannot communicate with each other at all, but can each communicate with a third (Root) port, a potential loop exists if one of the Designated Ports has a priority vector that is worse than that of the Root Port. To ensure that such loop does not occur, a Root Port that receives an inferior message from a Designated Port detects a dispute if the Learning flag is set, and transitions to Discarding.

13.20 In-service upgrades

It can be desirable to upgrade the control plane software of a bridging system, without interrupting existing data connectivity, while the Spanning Tree Protocol Entity is not operating. This can be done, without the risk of creating data loops, providing that the other bridges in the network are suitably configured. However the failure of a network component (bridge or LAN) while the upgrade is in progress can result in a loss of connectivity, as the ways in which the network can reconfigure are restricted. This clause (13.20) describes the necessary configuration of the other bridges in the network; the behavior of the upgrading system is assumed to be system dependent and is not specified in detail, except as follows:

- a) Frames can be received from and transmitted to ports that were forwarding prior to beginning the upgrade, and are not forwarded to or from any other port.
- b) Frames received on ports that were learning prior to beginning the upgrade can be submitted to the Learning Process, while frames received on other ports are not.
- c) BPDUs are not transmitted, and received BPDUs are discarded, while the upgrade is in progress.
- d) If there is to be no interruption in connectivity when the upgrade is complete, the parameters of BPDUs received prior to beginning the upgrade are retained.

NOTE 1—If received BPDU information is not retained, the spanning tree protocol variables and state machines are reinitialized by asserting BEGIN, and there will be a brief interruption in connectivity.

The need for a control plane software upgrade can result from the need to change any component of that software, and rarely from upgrades to the Spanning Tree Protocol Entity itself. The in-service upgrade procedures described here depend on the operation of specifically identified features of the RSTP and MSTP specified in this standard (Clause 13), and do not ensure loop-free operation if the necessary criteria are not met by all bridges in the network. In particular safe upgrades are not supported in networks including bridges operating STP as specified in the IEEE Std 802.1D, 1998 Edition, and earlier revisions. It is essential for the network administrator to base-line the network, i.e. verify its configuration and the configuration of its components, prior to attempting an in-service upgrade. It is recommended that a single system be upgraded at a time, and its subsequent correct operation verified prior to making further changes.

If all the LANs that connect the bridges in the network provide point-to-point connectivity, connecting at most two bridges, and each Bridge Port thus connecting to another bridge meets the following conditions:

- e) `operPointToPointMAC` (13.26.14) is TRUE; and
- f) `operEdge` (13.25.30) is FALSE;

NOTE 2—Apart from the requirement for a point-to-point physical topology, these conditions can be ensured by setting `adminPointToPointMAC` (6.6.3) TRUE, and both `AdminEdge` and `AutoEdge` FALSE.

then a Designated Port will not transition from Discarding to Learning or Forwarding without receiving an Agreement from its immediate neighbor (see 13.35). This prevents the introduction of data loops while one or more bridges are being upgraded, even if other bridges or LANs fail or are added to the network. To prevent existing connectivity being disrupted (provided no other network additions or failures occur) the following are also necessary:

- g) Prior to the upgrade, each of the upgrading bridge's immediately neighboring ports that is a Root Port (for a given tree) has to be configured as an Layer 2 Gateway Port (13.18) (for that tree), with the same information that it was previously receiving from the upgrading bridge.
- h) After the upgrade, each of the ports thus temporarily configured as an Layer 2 Gateway Port needs to have its normal configuration restored.

NOTE 3—If the upgrading bridge is capable of sending periodic ‘canned BPDUs’ containing the same information as immediately prior to the upgrade, there is no requirement for neighbor L2GP configuration, or for the point-to-point and `operEdge` conditions. Attempts by neighboring bridges to operate outside the parameters dictated by the ‘canned BPDUs’ will result in disputes, preventing new connectivity.

13.21 Fragile bridges

Some, nonconformant, bridging systems are known to be ‘fragile’, i.e. they can suffer from unpredictable interruptions to Spanning Tree Protocol Entity operation and will forward data frames while no longer sending or receiving BPDUs, on some or all ports. If the spanning tree protocol implementations of the other bridges in the network conform to prior revisions of this standard and IEEE Std 802.1D, these interruptions can result data loops even in networks of point-to-point LANs. This revision of this standard allows a Bridge

Port attached to point-to-point LAN to ensure that its neighbor remains capable of receiving and transmitting BPDUs, even if that neighbor's RSTP or MSTP implementation conforms to a prior revision of this standard, and to block connectivity (by transitioning to Discarding) otherwise. The CIST Proposal flag is set in all BPDUs transmitted by a Designated Port, and used to solicit an Agreement from the neighbor (which might otherwise not transmit). This capability is controlled by the AutoIsolate (13.25.6) variable, is disabled by default to allow for in-service upgrades, and is only effective if the neighboring bridge is capable of RSTP or MSTP operation.

13.22 Spanning tree protocol state machines

Each Spanning Tree Protocol Entity's operation of the protocols specified in this clause (Clause 13) is specified by the following state machines:

- a) Port Role Selection (PRS, 13.34)

for the CIST and for each MSTI, with the following state machines for each Bridge Port:

- b) Port Timers (PTI, 13.28)
- c) Port Protocol Migration (PPM, 13.30)
- d) Port Receive (PRX, 13.29)
- e) Port Transmit (PRT, 13.32)
- f) Bridge Detection (BDM, 13.31)

and the following optional state machine for each Bridge Port:

- g) Layer 2 Gateway Port Receive (L2GPRX, 13.38)

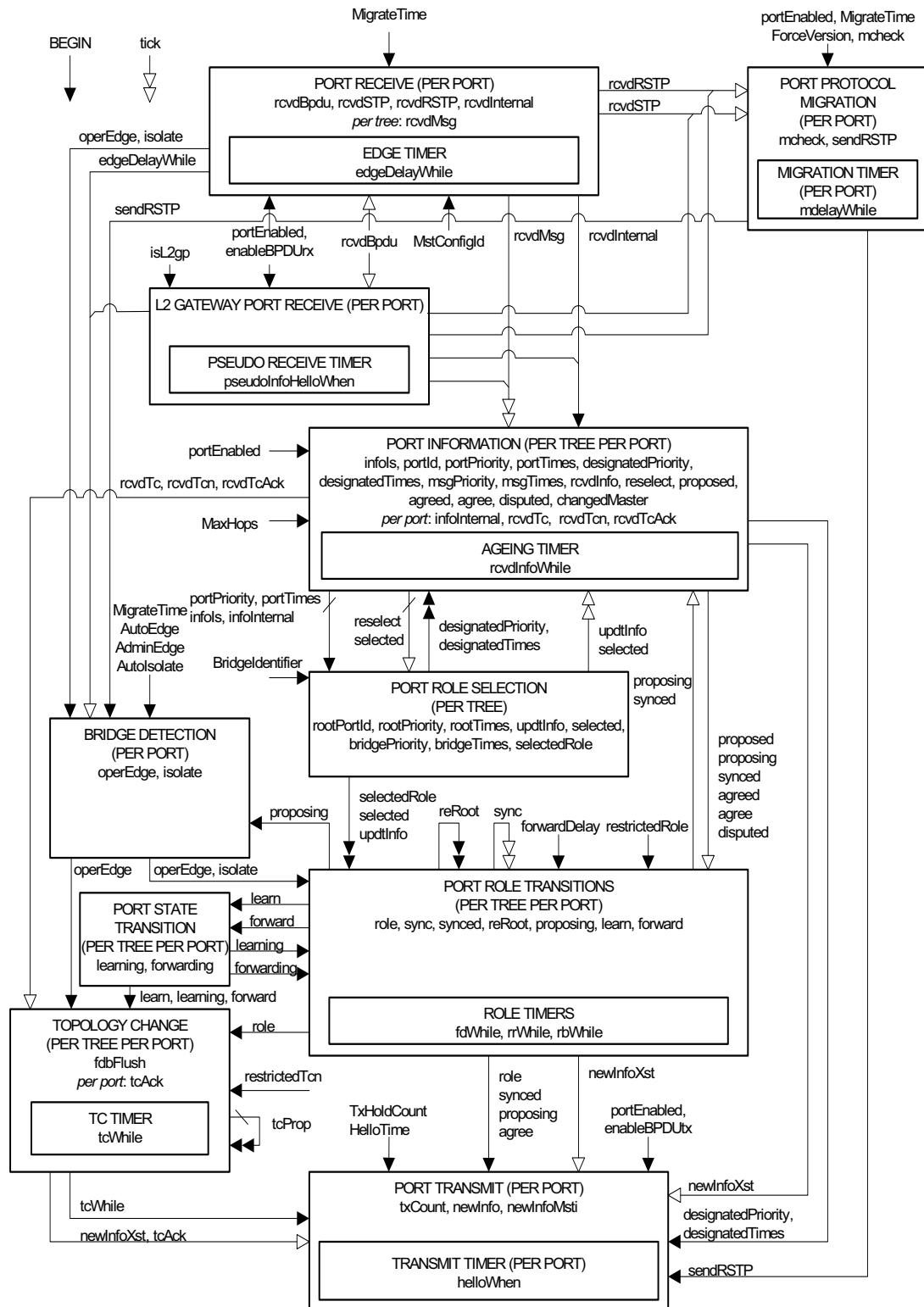
and the following state machines for each Bridge Port for the CIST and for each MSTI:

- h) Port Information (PIM, 13.33)
- i) Topology Change (TCM, 13.37)

and the following state machines for each Bridge Port for the CIST, for each MSTI:

- j) Port Role Transitions (PRT, 13.35)
- k) Port State Transition (PST, 13.36)

Each state machine and its associated variable and procedural definitions are specified in detail in 13.23 through 13.38. The state machine notation is specified in Annex E. Figure 13-13 is not itself a state machine but provides an overview of the state machines, their state variables, and communication between machines. Figure 13-14 describes its notation.



NOTE: For convenience all timers are collected together into one state machine.

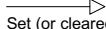
Figure 13-13—Spanning tree protocol state machines—overview and relationships

NOTATION:

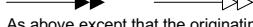
Variables are shown both within the machine where they are principally used and between machines where they are used to communicate information. In the latter case they are shown with a variety of arrow styles, running from one machine to another, that provide an overview of how the variables are typically used:



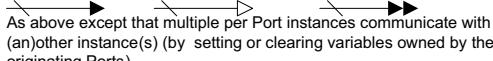
Not changed by the target machine. Where the state machines are both per Port, this variable communicates between machine instances for the same port.



Set (or cleared) by the originating machine, cleared (or set) by the target machine. Where the state machines are both per Port, this variable communicates between machine instances for the same port.



As above except that the originating per port machine instance communicates with multiple port machine instances (by setting or clearing variables owned by those Ports).



As above except that multiple per Port instances communicate with (an)other instance(s) (by setting or clearing variables owned by the originating Ports).

ABBREVIATIONS:

BDM:	Bridge Detection Machine
PIM:	Port Information Machine
PPM:	Port Protocol Migration Machine
PRS:	Port Role Selection Machine
PRT:	Port Role Transitions Machine
PRX:	Port Receive Machine
PST:	Port State Transitions Machine
PTI:	Port Timers Machine
PTX:	Port Transmit Machine
TCM:	Topology Change Machine

Figure 13-14—MSTP overview notation

13.23 State machine timers

The timer variables declared in this subclause are part of the specification. The accompanying descriptions are provided to aid in the comprehension of the protocol only, and are not part of the specification. Each timer variable represents an integral number of seconds before timer expiry.

One instance of the following shall be implemented per port:

- a) edgeDelayWhile (13.23.1)
- b) helloWhen (13.23.3)
- c) mdelayWhile (13.23.4)

One instance of the following shall be implemented per port when L2GP functionality is provided:

- d) pseudoInfoHelloWhen (13.23.10)

One instance per port of the following shall be implemented for the CIST and one per port for each MSTI:

- e) fdWhile (13.23.2)
- f) rrWhile (13.23.7)
- g) rbWhile (13.23.5)
- h) tcWhile (13.23.9)
- i) rcvdInfoWhile (13.23.6)
- j) tcDetected (13.23.8).

Table 13-5 specifies values and ranges for the initial values of timers and for transmission rate limiting performance parameters. Default values are specified to avoid the need to set values prior to operation in most cases, and are widely applicable to networks using the spanning tree protocols specified in this standard. The table recommends Bridge Hello Time, Bridge Max Age, and Bridge Forward Delay values that maximise interoperability in networks that include bridges using STP as specified in IEEE Std 802.1D, 1998 Edition. Ranges are specified to ensure that the protocols operate correctly.

NOTE—Changes to Bridge Forward Delay do not affect reconfiguration times, unless the network includes bridges that do not conform to this revision of this standard. Changes to Bridge Max Age can have an effect, as it is possible for old information to persist in loops in the physical topology for a number of “hops” equal to the value of Max Age in seconds, and thus exhaust the Transmit Hold Count in small loops.

Table 13-5—Timer and related parameter values

Parameter	Default	Permitted Range	Interoperability recommendations
Migrate Time	3.0	— ^a	— ^a
(Bridge) Hello Time	2.0	— ^a	— ^a
Bridge Max Age	20.0	6.0–40.0	20.0
Bridge Forward Delay	15.0	4.0–30.0	15.0
Transmit Hold Count	6	1–10	6
Max Hops	20	6–40	—

All times are in seconds. —^a Not applicable, value is fixed.

Bridge Max Age, Bridge Forward Delay, and Transmit Hold Count may be set by management, if this capability is provided the bridge shall have the capability to use the full range of values in the parameter ranges specified in the Permitted Range column of Table 13-5, with a timer resolution of r seconds, where $0 < r \leq 1$. To support interoperability with previous revisions of this standard and IEEE Std 802.1D, a bridge shall enforce the following relationships:

$$2 \times (\text{Bridge_Forward_Delay} - 1.0 \text{ seconds}) \geq \text{Bridge_Max_Age}$$

$$\text{Bridge_Max_Age} \geq 2 \times (\text{Bridge_Hello_Time} + 1.0 \text{ seconds})$$

13.23.1 edgeDelayWhile

The Edge Delay timer. The time remaining, in the absence of a received BPDU, before this port is identified as an operEdgePort.

13.23.2 fdWhile

The Forward Delay timer. Used to delay Port State transitions until other Bridges have received spanning tree information.

13.23.3 helloWhen

The Hello timer. Used to ensure that at least one BPDU is transmitted by a Designated Port in each HelloTime period.

13.23.4 mdelayWhile

The Migration Delay timer. Used by the Port Protocol Migration state machine to allow time for another RSTP Bridge on the same LAN to synchronize its migration state with this port before the receipt of a BPDU can cause this port to change the BPDU types it transmits. Initialized to MigrateTime (13.24.5).

13.23.5 rbWhile

The Recent Backup timer. Maintained at its initial value, twice HelloTime, while the port is a Backup Port.

13.23.6 rcvdInfoWhile

The Received Info timer. The time remaining before information, i.e. portPriority (13.25.33) and portTimes (13.25.34), received in a Configuration Message is aged out if a further message is not received.

13.23.7 rrWhile

The Recent Root timer.

13.23.8 tcDetected

The Topology Change timer for MRP application usage. ‘New’ messages are sent while this timer is running (see 10.2).

13.23.9 tcWhile

The Topology Change timer. TCN Messages are sent while this timer is running.

13.23.10 pseudoInfoHelloWhen

The Pseudo Info Hello When timer. Used by the Layer 2 Gateway Port Receive (L2GPRX, 13.38) state machine to prompt the simulated reception at HelloTime intervals of a BPDU (see 13.27.11).

13.24 Per bridge variables

The variables declared in this clause (13.24) are part of the specification. The accompanying descriptions are provided to aid in the comprehension of the protocol only, and are not part of the specification.

There is one instance per bridge component of the following variable(s):

- a) ForceProtocolVersion (13.24.4)
- b) TxHoldCount (13.24.10)
- c) MigrateTime (13.24.5)

One instance of the following shall be implemented per bridge component if MSTP is implemented:

- d) MstConfigId (13.24.6)

The above parameters ((a) through (d)) are not modified by the operation of the spanning tree protocols, but are treated as constants by the state machines. If ForceProtocolVersion or MSTConfigId are modified by management, BEGIN shall be asserted for all state machines.

There is one instance per bridge of each of the following for the CIST, and one for each MSTI.

- e) Bridgelidentifer (13.24.1)
- f) BridgePriority (13.24.2)
- g) BridgeTimes (13.24.3)
- h) rootPortId (13.24.7)
- i) rootPriority (13.24.8)
- j) rootTimes (13.24.9)

Bridgelidentifer, BridgePriority, and BridgeTimes are not modified by the operation of the spanning tree protocols but are treated as constants by the state machines. If they are modified by management, spanning tree priority vectors and Port Role assignments for shall be recomputed, as specified by the operation of the Port Role Selection state machine (13.34) by clearing selected (13.25) and setting reselect (13.25) for all Bridge Ports for the relevant MSTI and for all trees if the CIST parameter is changed.

13.24.1 BridgeIdentifier

The unique Bridge Identifier assigned to this bridge for this tree (CIST or MSTI).

The 12-bit system ID extension component of a Bridge Identifier (14.2.5) shall be set to zero for the CIST, and to the value of the MSTID for an MSTI, thus allocating distinct Bridge Identifiers to the CIST and each MSTI—all based on the use of a single Bridge Address component value for the MST Bridge as a whole.

NOTE—This convention is used to convey the MSTID for each MSTI Configuration Message in an MST BPDU.

The 4 most significant bits of the Bridge Identifier (the settable Priority component) for the CIST and for each MSTI can be modified independently of the setting of those bits for all other trees, as a part of allowing full and independent configuration control to be exerted over each Spanning Tree instance.

13.24.2 BridgePriority

For the CIST, the value of the CIST bridge priority vector, as defined in 13.9. The CIST Root Identifier, CIST Regional Root Identifier, and Designated Bridge Identifier components are all equal to the value of the CIST Bridge Identifier. The remaining components (External Root Path Cost, Internal Root Path Cost, and Designated Port Identifier) are set to zero.

For a given MSTI, the value of the MSTI bridge priority vector, as defined in 13.10. The MSTI Regional Root Identifier and Designated Bridge Identifier components are equal to the value of the MSTI Bridge Identifier (13.24.1). The remaining components (MSTI Internal Root Path Cost, Designated Port Identifier) are set to zero.

BridgePriority is used by `updtrolesTree()` in determining the value of the `rootPriority` variable (see 13.24.8).

13.24.3 BridgeTimes

For the CIST, BridgeTimes comprises:

- a) The current values of Bridge Forward Delay and Bridge Max Age (13.23, Table 13-5).
- b) A Message Age value of zero.
- c) The current value of Max Hops (13.24.3). This parameter value is determined only by management.

For a given MSTI, BridgeTimes comprises:

- d) The current value of MaxHops (Max Hops in Table 13-5), the initial value of remainingHops for MSTI information generated at the boundary of an MSTI region (see 13.24.9).

BridgeTimes is used by `updtrolesTree()` in determining the value of the `rootTimes` variable (13.24.9).

13.24.4 ForceProtocolVersion

The Force Protocol Version parameter for the Bridge (13.6.2).

13.24.5 MigrateTime

The value of the Migrate Time parameter as specified in Table 13-5. This value shall not be changed.

13.24.6 MstConfigId

The current value of the bridge's MST Configuration Identifier (13.7). Changes in this parameter cause BEGIN to be asserted for the state machines for the bridge, for all trees, and for each port.

13.24.7 rootPortId

For the CIST, the Port Identifier of the Root Port, and a component of the CIST root priority vector (13.9).

For a given MSTI, the Port Identifier of the Root Port, and a component of the MSTI root priority vector (13.10).

13.24.8 rootPriority

For the CIST: the Root Identifier, External Root Path Cost, Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the bridge's CIST root priority vector (13.9).

For a given MSTI: the MSTI Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the bridge's MSTI root priority vector (13.10).

13.24.9 rootTimes

For the CIST, the Bridge's timer parameter values (Message Age, Max Age, Forward Delay, and remainingHops). The values of these timers are derived (see 13.27.31) from the values stored in the CIST's portTimes parameter (13.25.34) for the Root Port or from BridgeTimes (13.24.3).

For a given MSTI, the value of remainingHops derived (13.27.31) from the value stored in the MSTI's portTimes parameter (13.25.34) for the Root Port or from BridgeTimes (13.24.3).

13.24.10 TxHoldCount

The value of Transmit Hold Count (Table 13-5) for the bridge. If this is modified, the value of txCount (13.25) for all ports shall be set to zero.

13.25 Per port variables

The variables declared in this clause (13.25) are part of the specification. The accompanying descriptions are provided to aid in the comprehension of the protocol only, and are not part of the specification.

There is one instance per port of each of the following variables:

- a) AdminEdge (13.25.1)
- b) ageingTime (13.25.2)
- c) AutoEdge (13.25.5)
- d) Autolsolate (13.25.6)
- e) enableBPDUrx (13.25.10)
- f) enableBPDUTx (13.25.11)
- g) isL2gp (13.25.19)
- h) isolate (13.25.20)
- i) mcheck (13.25.25)
- j) newInfo (13.25.28)
- k) operEdge (13.25.30)

- l) portEnabled (13.25.31)
- m) rcvdBpdu (13.25.38)
- n) rcvdRSTP (13.25.42)
- o) rcvdSTP (13.25.43)
- p) rcvdTcAck (13.25.45)
- q) rcvdTcn (13.25.46)
- r) restrictedRole (13.25.49)
- s) restrictedTcn (13.25.50)
- t) sendRSTP (13.25.54)
- u) tcAck (13.25.57)
- v) tick (13.25.59)
- w) txCount (13.25.60)

If MSTP is implemented there is one instance per port, applicable to the CIST and to all MSTIs, of the following variable(s):

- x) rcvdInternal (13.25.40)

If MSTP is implemented there is one instance per port of each of the following variables for the CIST:

- y) ExternalPortPathCost (13.25.12)
- z) infolnernal (13.25.16)
- aa) master (13.25.23)
- ab) mastered (13.25.24)

A single per port instance of the following variable(s) applies to all MSTIs:

- ac) newInfoMsti (13.25.29)

If the Layer 2 Gateway Port Receive state machine (13.18, 13.38) is implemented for a port there is one instance for the CIST, and one instance for each MSTI, for that port of the following variable:

- ad) pseudoRootId (13.25.37)

There is one instance per port of each of the following variables for the CIST, one per port for each MSTI:

- ae) agree (13.25.3)
- af) agreed (13.25.4)
- ag) designatedPriority (13.25.7)
- ah) designatedTimes (13.25.8)
- ai) disputed (13.25.9)
- aj) fdbFlush (13.25.13)
- ak) forward (13.25.14)
- al) forwarding (13.25.15)
- am) infols (13.25.17)
- an) InternalPortPathCost (13.25.18)
- ao) learn (13.25.21)
- ap) learning (13.25.13)
- aq) msgPriority (13.25.26)
- ar) msgTimes (13.25.27)
- as) portId (13.25.32)
- at) portPriority (13.25.33)
- au) portTimes (13.25.34)
- av) proposed (13.25.35)

- aw) proposing (13.25.36)
- ax) rcvdInfo (13.25.39)
- ay) rcvdMsg (13.25.41)
- az) rcvdTc (13.25.44)
- ba) reRoot (13.25.47)
- bb) reselect (13.25.48)
- bc) role (13.25.51)
- bd) selected (13.25.52)
- be) selectedRole (13.25.53)
- bf) sync (13.25.55)
- bg) synced (13.25.56)
- bh) tcProp (13.25.58)
- bi) updtInfo (13.25.61)

13.25.1 AdminEdge

A Boolean. Set by management if the port is to be identified as *operEdge* immediately on initialization, without a delay to detect other bridges attached to the LAN. The recommended default value is FALSE.

13.25.2 ageingTime

Filtering database entries for this port are aged out after *ageingTime* has elapsed since they were first created or refreshed by the Learning Process. The value of this parameter is normally Ageing Time (8.8.3, Table 8-6), and is changed to *FwdDelay* (13.26.8) for a period of *FwdDelay* after *fdbFlush* (13.25.13) is set by the topology change state machine if *stpVersion* (13.26.20) is TRUE.

13.25.3 agree

A Boolean. See 13.15 and Figure 13-12.

13.25.4 agreed

A Boolean. Set, for the CIST or an MSTI, if an Agreement has been received indicating that the Port States and transmitted priority vectors for the other bridge attached to this LAN are (and, in the absence of further communication with this bridge and within the design probabilities of protocol failure due to repeated BPDU loss, will remain) compatible with a loop-free active topology determined by this port's priority vectors (13.15, 13.22).

13.25.5 AutoEdge

A Boolean. Set by management if the bridge detection state machine (BDM, 13.31) is to detect other bridges attached to the LAN, and set *operEdge* automatically. The recommended default is TRUE.

13.25.6 Autolsolate

A Boolean. Set by management if *isolate* (13.25.20) is to be set, causing a Designated Port to transition to Discarding if both *AdminEdge* (13.25.1) and *AutoEdge* (13.25.5) are FALSE and the other bridge presumed to be attached to the same point-to-point LAN does not transmit periodic BPDUs—either as a Designated Port or in response to BPDUs with the Proposal flag set (see 13.21). The recommended default of this parameter is FALSE. *AdminEdge* and *AutoEdge* are both reset only on ports that are known to connect to other bridges.

13.25.7 designatedPriority

For the CIST and a given port, the CIST Root Identifier, External Root Path Cost, Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the port's CIST designated priority vector, as defined in 13.9.

For a given MSTI and port, the Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the designated priority vector, as defined in 13.10.

13.25.8 designatedTimes

For the CIST and a given port, the set of timer parameter values (Message Age, Max Age, Forward Delay, and remainingHops) that are used to update Port Times when updtInfo is set. These timer parameter values are used in BPDUs transmitted from the port. The value of designatedTimes is copied from the CIST rootTimes Parameter (13.24.9) by the operation of the updtRolesTree() procedure.

For a given MSTI and port, the value of remainingHops used to update this MSTI's portTimes parameter when updtInfo is set. This timer parameter value is used in BPDUs transmitted from the port. The updtRolesTree() procedure (13.27.31) copies designatedTimes from the MSTI's rootTimes (13.24.9).

13.25.9 disputed

A Boolean. See 13.19, 13.18, Figure 13-20, 13.27.15, Figure 13-24, and Figure 13-25.

13.25.10 enableBPDURx

A Boolean. This per port management parameter is set by default, and should not be clear unless the port is configured as a Layer Two Gateway Port (i.e. isL2gp is set). When clear it can allow loops to be created, or can result in no connectivity. When cleared, BPDUs received on the port are discarded and not processed.

13.25.11 enableBPDUtx

A Boolean. This per port management parameter is set by default, and should not be clear unless the port is configured as a Layer Two Gateway Port (i.e. isL2gp is set). When clear it can allow loops to be created, or can result in no connectivity. When cleared, no BPDUs are transmitted by this port.

13.25.12 ExternalPortPathCost

The port's contribution, when it is the CIST Master Port (for MSTP) or the CST Root Port (for RSTP), to the External Root Path Cost (13.8) for the bridge.

13.25.13 fdbFlush

A Boolean. Set by the topology change state machine to instruct the filtering database to remove entries for this port, immediately if rstpVersion (13.26.19) is TRUE, or by rapid ageing (13.25.2) if stpVersion (13.26.20) is TRUE. Reset by the filtering database once the entries are removed if rstpVersion is TRUE, and immediately if stpVersion is TRUE. Setting the fdbFlush variable does not result in removal of filtering database entries in the case that the port is an Edge Port (i.e., operEdge is TRUE). The filtering database removes entries only for those VIDs that have a fixed registration (see 10.7.2) on any port of the bridge that is not an Edge Port.

NOTE—If MVRP is in use, the topology change notification and flushing mechanisms defined in MRP (Clause 10) and MVRP (11.2.5) are responsible for filtering entries in the Filtering Database for VIDs that are dynamically registered using MVRP (i.e., for which there is no fixed registration in the bridge on non-Edge Ports).

13.25.14 forward

A Boolean. Set or cleared by the Port Role Transitions state machine (13.35) to instruct the Port State Transitions state machine (13.36) to enable or disable forwarding.

13.25.15 forwarding

A Boolean. Set or cleared by the Port Port State Transitions state machine (13.36) to indicate that forwarding has been enabled or disabled.

13.25.16 infolInternal

If infols is Received, indicating that the port has received current information from the Designated Bridge for the attached LAN, infolInternal is set if that Designated Bridge is in the same MST Region as the receiving bridge and reset otherwise.

13.25.17 infols

A variable that takes the values Mine, Aged, Received, or Disabled, to indicate the origin/state of the port's Spanning Tree information (portInfo) held for the port, as follows:

- a) If infols is Received, the port has received current (not aged out) information from the Designated Bridge for the attached LAN (a point-to-point bridge link being a special case of a LAN).
- b) If infols is Mine, information for the port has been derived from the Root Port for the Bridge (with the addition of root port cost information). This includes the possibility that the Root Port is "Port 0," i.e., the bridge is the Root Bridge for the Bridged Local Area Network.
- c) If infols is Aged, information from the Root Bridge has been aged out. Just as for reselect (13.25.48), the state machine does not formally allow the Aged state to persist. However, if there is a delay in recomputing the new root port, correct processing of a received BPDU is specified.
- d) Finally if the port is disabled, infols is Disabled.

13.25.18 InternalPortPathCost

The port's contribution, when it is an IST Root Port (for MSTP) to the Internal Root Path Cost (13.8) for the bridge.

13.25.19 isL2gp

A Boolean. Set by management to identify a port configured to be a Layer Two Gateway Port. This parameter is set to FALSE by default. When set, enableBPDUTx should be cleared.

13.25.20 isolate

A Boolean. Set by the bridge detection state machine (BDM, 13.31) when the Spanning Tree Protocol Entity of a neighboring bridge has apparently failed (see 13.21, 13.25.6).

13.25.21 learn

A Boolean. Set or cleared by the Port Role Transitions state machine (13.35) to instruct the Port State Transitions state machine (13.36) to enable or disable learning.

13.25.22 learning

A Boolean. Set or cleared by the Port Port State Transitions state machine (13.36) to indicate that learning has been enabled or disabled.

13.25.23 master

A Boolean. Used to determine the value of the Master flag for a given MSTI and port in transmitted MST BPDUs.

Set TRUE if the Port Role for the MSTI and Port is Root Port or Designated Port, and the bridge has selected one of its ports as the Master Port for this MSTI or the mastered flag is set for this MSTI for any other Bridge Port with a Root Port or Designated Port Role. Set FALSE otherwise.

13.25.24 mastered

A Boolean. Used to record the value of the Master flag for a given MSTI and port in MST BPDUs received from the attached LAN.

NOTE—**master** and **mastered** signal the connection of the MSTI to the CST via the Master Port throughout the MSTI. These variables and their supporting procedures do not affect the connectivity provided by this standard but permit future enhancements to MSTP providing increased flexibility in the choice of Master Port without abandoning plug-and-play network migration. They are, therefore, omitted from the overviews of protocol operation, including Figure 13-13.

13.25.25 mcheck

A Boolean. May be set by management to force the Port Protocol Migration state machine to transmit RST (or MST) BPDUs for a MigrateTime (13.24.5, Table 13-5) period, to test whether all STP Bridges on the attached LAN have been removed and the port can continue to transmit RSTP BPDUs. Setting mcheck has no effect if stpVersion (13.26.20) is TRUE.

13.25.26 msgPriority

For the CIST and a given port, the CIST Root Identifier, External Root Path Cost, Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the CIST message priority vector conveyed in a received BPDU, as defined in 13.9.

For a given MSTI and port, the Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the MSTI message priority vector, as defined in 13.10 and conveyed in a received BPDU for this MSTI.

13.25.27 msgTimes

For the CIST and a given port, the timer parameter values (Message Age, Max Age, Forward Delay, Hello Time, and remainingHops) conveyed in a received BPDU. If the BPDU is an STP or RST BPDU without MSTP parameters, remainingHops is set to the value of the MaxHops component of BridgeTimes (13.24.3).

For a given MSTI and port, the value of remainingHops received in the same BPDU as the message priority components of this MSTI's msgPriority parameter.

13.25.28 newInfo

A Boolean. Set TRUE if a BPDU conveying changed CIST information is to be transmitted. It is set FALSE by the Port Transmit state machine.

13.25.29 newInfoMsti

A Boolean. Set TRUE if a BPDU conveying changed MSTI information is to be transmitted. It is set FALSE by the Port Transmit state machine.

13.25.30 operEdge

A Boolean. The value of the operEdgePort parameter [item l) in 12.8.2.1.3], as determined by the operation of the Bridge Detection state machine (13.31).

13.25.31 portEnabled

A Boolean. Set if the Bridge's MAC Relay Entity and Spanning Tree Protocol Entity can use the MAC Service provided by the Bridge Port's MAC entity to transmit and receive frames to and from the attached LAN, i.e., portEnabled is TRUE if and only if:

- a) MAC_Operational (6.6.2) is TRUE; and
- b) Administrative Bridge Port State (8.4, 13.11) for the port is Enabled.

13.25.32 portId

The Port Identifier for this port. This variable forms a component of the port priority and designated priority vectors (13.9,13.10).

The 4 most significant bits of the Port Identifier (the settable Priority component) for the CIST and for each MSTI can be modified independently of the setting of those bits for all other trees, as a part of allowing full and independent configuration control to be exerted over each Spanning Tree instance.

13.25.33 portPriority

For the CIST and a given port, the CIST Root Identifier, External Root Path Cost, Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the port's port priority vector (13.9).

For a given MSTI and port, the Regional Root Identifier, Internal Root Path Cost, Designated Bridge Identifier, and Designated Port Identifier components of the port's MSTI port priority vector (13.10).

13.25.34 portTimes

For the CIST and a given port, the port's timer parameter values (Message Age, Max Age, Forward Delay, Hello Time, and remainingHops). The Hello Time timer parameter value is used in transmitted BPDUs.

For a given MSTI and port, the value of remainingHops for this MSTI in transmitted BPDUs.

13.25.35 proposed

A Boolean. See 13.15, Figure 13-10, Figure 13-12, 13.27.14, 13.27.18, 13.35, and Figure 13-24.

13.25.36 proposing

A Boolean. See 13.15, Figure 13-10, Figure 13-12, 13.27.14, 13.27.14, 13.31, and Figure 13-25.

13.25.37 pseudoRootId

A Bridge Identifier configured by management for L2GP operation. By default, it is set to the Bridgeln identifier (13.24.1).

13.25.38 rcvdBPDU

A Boolean. Set by system dependent processes, this variable notifies the Port Receive state machine (13.29) when a valid (Clause 14) Configuration, TCN, RST, or MST BPDU (14.3) is received on the port. Reset by the Port Receive state machine.

13.25.39 rcvdInfo

Set to the result of the rcvInfo() procedure (13.27.12).

13.25.40 rcvdInternal

A Boolean. Set TRUE by the Receive Machine if the BPDU received was transmitted by a bridge in the same MST Region as the receiving bridge.

13.25.41 rcvdMsg

A Boolean. See 13.27.13, and 13.29.

13.25.42 rcvdRSTP

A Boolean. See 13.29, 13.27.29, and 13.30.

13.25.43 rcvdSTP

A Boolean. See 13.29, 13.27.29, and 13.30.

13.25.44 rcvdTc

A Boolean. See 13.27.13, 13.27.23 and 13.37.

13.25.45 rcvdTcAck

A Boolean. See 13.27.23 and 13.37.

13.25.46 rcvdTcn

A Boolean. See 13.27.13, 13.27.23 and 13.37.

13.25.47 reRoot

A Boolean. Set by a newly selected Root Port to force prior Root Ports to Discarding, before it becomes forwarding. See Figure 13-24, 13.27.20, Figure 13-23, Figure 13-25, and Figure 13-26.

13.25.48 reselect

A Boolean. Set to prompt recomputation of the CIST or an MSTI. See 13.33 and 13.34.

13.25.49 restrictedRole

A Boolean. Set by management. If TRUE causes the port not to be selected as Root Port for the CIST or any MSTI, even it has the best spanning tree priority vector. Such a port will be selected as an Alternate Port after the Root Port has been selected. This parameter should be FALSE by default. If set, it can cause lack of spanning tree connectivity. It is set by a network administrator to prevent bridges external to a core region of the network influencing the spanning tree active topology, possibly because those bridges are not under the full control of the administrator.

13.25.50 restrictedTcn

A Boolean. Set by management. If TRUE causes the port not to propagate received topology change notifications and topology changes to other ports. This parameter should be FALSE by default. If set it can cause temporary loss of connectivity after changes in a spanning trees active topology as a result of persistent incorrectly learned station location information. It is set by a network administrator to prevent bridges external to a core region of the network, causing address flushing in that region, possibly because those bridges are not under the full control of the administrator or MAC_Operational for the attached LANs transitions frequently.

13.25.51 role

The current Port Role. DisabledPort, RootPort, DesignatedPort, AlternatePort, BackupPort, or MasterPort.

NOTE—The MasterPort role applies to each MSTI when the CIST Port Role is RootPort and connects to another MST Region. An MSTI Master Port is part of the stable active topology for frames assigned to that MSTI, just as the CIST Root Port forwards frames for the IST. The Port State for each MSTI may differ as required to suppress temporary loops.

13.25.52 selected

A Boolean. See 13.34, 13.27.21.

13.25.53 selectedRole

A newly computed role for the port.

13.25.54 sendRSTP

A Boolean. See 13.30, 13.32.

13.25.55 sync

A Boolean. Set to force the Port State to be compatible with the loop-free active topology determined by the priority vectors held by this bridge (13.15, 13.22) for this tree (CIST, or MSTI), transitioning the Port State to Discarding, and soliciting an Agreement if possible, if the port is not already synchronized (13.25.56).

13.25.56 synced

A Boolean. TRUE only if the Port State is compatible with the loop-free active topology determined by the priority vectors held by this bridge for this tree (13.15, 13.22).

13.25.57 tcAck

A Boolean. Set to transmit a Configuration Message with a topology change acknowledge flag set.

13.25.58 tcProp

A Boolean. Set by the Topology Change state machine of any other port, to indicate that a topology change should be propagated through this port.

13.25.59 tick

A Boolean. See the Port Timers state machine (13.28).

13.25.60 txCount

A counter. Incremented by the Port Transmission (13.32) state machine on every BPDU transmission, and decremented used by the Port Timers state machine (13.28) once a second. Transmissions are delayed if txCount reaches TxHoldCount (13.24.10).

13.25.61 updtnfo

A boolean. Set by the Port Role Selection state machine (13.34, 13.27.31) to tell the Port Information state machine that it should copy designatedPriority to portPriority and designatedTimes to portTimes.

13.26 State machine conditions and parameters

The following variable evaluations are defined for notational convenience in the state machines:

- a) allSynced (13.26.1)
- b) allTransmitReady (13.26.2)
- c) cist (13.26.3)
- d) cistRootPort (13.26.4)
- e) cistDesignatedPort (13.26.5)
- f) EdgeDelay (13.26.6)
- g) forwardDelay (13.26.7)
- h) FwdDelay (13.26.8)
- i) HelloTime (13.26.9)
- j) MaxAge (13.26.11)
- k) msti (13.26.10)
- l) mstiDesignatedOrTCpropagatingRootPort (13.26.12)
- m) mstiMasterPort (13.26.13)
- n) operPointToPointMAC (13.26.14)
- o) rcvdAnyMsg (13.26.15)
- p) rcvdCistMsg (13.26.16)
- q) rcvdMstiMsg (13.26.17)
- r) reRooted (13.26.18)
- s) rstpVersion (13.26.19)
- t) stpVersion (13.26.20)
- u) updtnfo (13.26.21)
- v) updtnfo (13.26.22)

13.26.1 allSynced

The condition allSynced is TRUE for a given port, for a given tree, if and only if

- a) For all ports for the given tree, selected is TRUE, the port's role is the same as its selectedRole, and updtnfo is FALSE; and

- b) The role of the given port is
 - 1) Root Port or Alternate Port and `synced` is TRUE for all ports for the given tree other than the Root Port; or
 - 2) Designated Port and `synced` is TRUE for all ports for the given tree other than the given port; or
 - 3) Master Port and `synced` is TRUE for all ports for the given tree other than the given port.

13.26.2 `allTransmitReady`

TRUE, if and only if, for the given port for all trees

- a) `selected` is TRUE; and
- b) `updInfo` is FALSE.

13.26.3 `cist`

TRUE only for CIST state machines; i.e., FALSE for MSTI state machine instances.

13.26.4 `cistRootPort`

TRUE if the CIST role for the given port is `RootPort`.

13.26.5 `cistDesignatedPort`

TRUE if the CIST role for the given port is `DesignatedPort`.

13.26.6 `EdgeDelay`

Returns the value of `MigrateTime` if `operPointToPointMAC` is TRUE, and the value of `MaxAge` otherwise.

13.26.7 `forwardDelay`

Returns the value of `HelloTime` if `sendRSTP` is TRUE, and the value of `FwdDelay` otherwise.

13.26.8 `FwdDelay`

The Forward Delay component of the CIST's `designatedTimes` parameter (13.25.8).

13.26.9 `HelloTime`

The Hello Time component of the CIST's `portTimes` parameter (13.25.34) with the recommended default value given in Table 13-5.

13.26.10 `msti`

TRUE only for MSTI state machines; i.e., FALSE for CIST state machine instances.

13.26.11 `MaxAge`

The Max Age component of the CIST's `designatedTimes` parameter (13.25.8).

13.26.12 `mstiDesignatedOrTCpropagatingRootPort`

TRUE if the role for any MSTI for the given port is either:

- a) DesignatedPort; or
- b) RootPort, and the instance for the given MSTI and port of the tcWhile timer is not zero.

13.26.13 mstiMasterPort

TRUE if the role for any MSTI for the given port is MasterPort.

13.26.14 operPointToPoint

TRUE if operPointToPointMAC (6.6.3) is TRUE for the Bridge Port.

13.26.15 rcvdAnyMsg

TRUE for a given port if rcvdMsg is TRUE for the CIST or any MSTI for that port.

13.26.16 rcvdCistMsg

TRUE for a given port if and only if rcvdMsg is TRUE for the CIST for that port.

13.26.17 rcvdMstiMsg

TRUE for a given port and MSTI if and only if rcvdMsg is FALSE for the CIST for that port and rcvdMsg is TRUE for the MSTI for that port.

13.26.18 reRooted

TRUE if the rrWhile timer is clear (zero) for all Ports for the given tree other than the given Port.

13.26.19 rstpVersion

TRUE if ForceProtocolVersion (13.6.2) is greater than or equal to 2.

13.26.20 stpVersion

TRUE if Force Protocol Version (13.6.2) is less than 2.

13.26.21 updCistInfo

TRUE for a given port if and only if updInfo is TRUE for the CIST for that port.

13.26.22 updMstiInfo

TRUE for a given port and MSTI if and only if updInfo is TRUE for the MSTI for that port or updInfo is TRUE for the CIST for that port.

NOTE—The dependency of rcvdMstiMsg and updMstiInfo on CIST variables for the port reflects the fact that MSTIs exist in a context of CST parameters. The state machines ensure that the CIST parameters from received BPDUs are processed and updated prior to processing MSTI information.

13.27 State machine procedures

The following procedures perform the functions specified for both the state machines for all trees, or specifically for the CIST, or a given MSTI:

- a) betterOrSameInfo(newInfoIs) (13.27.1)
- b) clearAllRcvdMsgs() (13.27.2)
- c) clearReselectTree() (13.27.3)
- d) disableForwarding (13.27.4)
- e) disableLearning (13.27.5)
- f) enableForwarding (13.27.6)
- g) enableLearning (13.27.7)
- h) fromSameRegion() (13.27.8)
- i) newTcDetected() (13.27.9)
- j) newTcWhile() (13.27.10)
- k) pseudoRcvMsgs() (13.27.11)
- l) rcvInfo() (13.27.12)
- m) rcvMsgs() (13.27.13)
- n) recordAgreement() (13.27.14)
- o) recordDispute() (13.27.15)
- p) recordMastered() (13.27.16)
- q) recordPriority() (13.27.17)
- r) recordProposal() (13.27.18)
- s) recordTimes() (13.27.19)
- t) setReRootTree (13.27.20)
- u) setSelectedTree() (13.27.21)
- v) setSyncTree() (13.27.22)
- w) setTcFlags() (13.27.23)
- x) setTcPropTree() (13.27.24)
- y) syncMaster() (13.27.25)
- z) txConfig() (13.27.26)
- aa) txRstp() (13.27.27)
- ab) txTcn() (13.27.28)
- ac) updtBPDUVersion() (13.27.29)
- ad) updtRcvdInfoWhile() (13.27.30)
- ae) updtRolesTree() (13.27.31)
- af) updtRolesDisabledTree() (13.27.32)

All references to named variables in the specification of procedures are to instances of the variables corresponding to the instance of the state machine using the function, i.e., to the CIST or the given MSTI as appropriate. References to forwarding and learning apply to frames assigned to the specified tree.

13.27.1 betterOrSameInfo(newInfoIs)

Returns TRUE if, for a given port and tree (CIST, or MSTI), either

- a) The procedure's parameter newInfoIs is Received, and infoIs is Received and the msgPriority vector is better than or the same as (13.9) the portPriority vector; or,
- b) The procedure's parameter newInfoIs is Mine, and infoIs is Mine and the designatedPriority vector is better than or the same as (13.9) the portPriority vector.

Returns False otherwise.

NOTE—This procedure is not invoked (in the case of a MSTI) if the received BPDU carrying the MSTI information was received from another MST Region. In that event, the Port Receive Machine (using rcvMsgs()) does not set rcvdMsg for any MSTI, and the Port Information Machine's SUPERIOR_DESIGNATED state is not entered.

13.27.2 clearAllRcvdMsgs()

Clears rcvdMsg for the CIST and all MSTIs, for this port.

13.27.3 clearReselectTree()

Clears reselect for the tree (the CIST or a given MSTI) for all ports of the bridge.

13.27.4 disableForwarding()

An implementation dependent procedure that causes the Forwarding Process (8.6) to stop forwarding frames through the port. The procedure does not complete until forwarding has stopped.

13.27.5 disableLearning()

An implementation dependent procedure that causes the Learning Process (8.7) to stop learning from the source address of frames received on the port. The procedure does not complete until learning has stopped.

13.27.6 enableForwarding()

An implementation dependent procedure that causes the Forwarding Process (8.6) to start forwarding frames through the port. The procedure does not complete until forwarding has been enabled.

13.27.7 enableLearning()

An implementation dependent procedure that causes the Learning Process (8.7) to start learning from frames received on the port. The procedure does not complete until learning has been enabled.

13.27.8 fromSameRegion()

Returns TRUE if rcvdRSTP is TRUE, and the received BPDU conveys a MST Configuration Identifier that matches that held for the bridge. Returns FALSE otherwise.

13.27.9 newTcDetected()

If the value of tcDetected is zero and sendRSTP is TRUE, this procedure sets the value of tcDetected to HelloTime plus one second. The value of HelloTime is taken from the CIST's portTimes parameter (13.25.34) for this port.

If the value of tcDetected is zero and sendRSTP is FALSE, this procedure sets the value of tcDetected to the sum of the Max Age and Forward Delay components of rootTimes.

Otherwise the procedure takes no action.

13.27.10 newTcWhile()

If the value of tcWhile is zero and sendRSTP is TRUE, this procedure sets the value of tcWhile to HelloTime plus one second and sets either newInfo TRUE for the CIST or newInfoMsti TRUE for a given MSTI. The value of HelloTime is taken from the CIST's portTimes parameter (13.25.34) for this port.

If the value of tcWhile is zero and sendRSTP is FALSE, this procedure sets the value of tcWhile to the sum of the Max Age and Forward Delay components of rootTimes and does not change the value of either newInfo or newInfoMsti.

Otherwise the procedure takes no action.

13.27.11 pseudoRcvMsgs()

Using local parameters, this procedure simulates the processing that would be applied by `rcvInfo()` and `rcvMsgs()` to a BPDU received on the port, from the same region and with the following parameters:

- a) Message Age, Max Age, Hello Time and Forward Delay are derived from `BridgeTimes` (13.24.3);
- b) The CIST information carries the message priority vector (13.9) with a value of {`pseudoRootId`, 0, `pseudoRootId`, 0, 0, 0};
- c) A CIST Port Role of Designated Port, with the Learning and Forwarding flags set;
- d) The Version 1 Length is 0 and Version 3 Length calculated appropriately;
- e) For each MSTI configured on the bridge, the corresponding MSTI Configuration Message carries:
 - 1) A message priority vector with a value of {`pseudoRootId`, 0, 0, 0};
 - 2) A Port Role of Designated Port, with the Learning and Forwarding flags set;
 - 3) MSTI Remaining Hops set to the value of the `MaxHops` component of `BridgeTimes` (13.24.3).

NOTE—If two L2GP ports are configured with the same CIST `pseudoRootId` then the IST may partition within the MST Region, but either of the L2GP ports can be selected to provide connectivity from the Region/customer network to a provider's network on an MSTI by MSTI basis.

13.27.12 rcvInfo()

Returns `SuperiorDesignatedInfo` if, for a given port and tree (CIST, or MSTI):

- a) The received CIST or MSTI message conveys a Designated Port Role, and
 - 1) The message priority (`msgPriority`—13.25.26) is superior (13.9 or 13.10) to the port's port priority vector, or
 - 2) The message priority is the same as the port's port priority vector, and any of the received timer parameter values (`msgTimes`—13.25.27) differ from those already held for the port (`portTimes`—13.25.34).

Otherwise, returns `RepeatedDesignatedInfo` if, for a given port and tree (CIST, or MSTI):

- b) The received CIST or MSTI message conveys a Designated Port Role, and
 - 1) A message priority vector and timer parameters that are the same as the port's port priority vector and timer values; and
 - 2) `infols` is Received.

Otherwise, returns `InferiorDesignatedInfo` if, for a given port and tree (CIST, or MSTI):

- c) The received CIST or MSTI message conveys a Designated Port Role.

Otherwise, returns `InferiorRootAlternateInfo` if, for a given port and tree (CIST, or MSTI):

- d) The received CIST or MSTI message conveys a Root Port, Alternate Port, or Backup Port Role and a CIST or MSTI message priority that is the same as or worse than the CIST or MSTI port priority vector.

Otherwise, returns `OtherInfo`.

NOTE—A Configuration BPDU implicitly conveys a Designated Port Role.

13.27.13 rcvMsgs()

This procedure is invoked by the Port Receive state machine (13.29) to decode a received BPDU. Sets rcvdTcn and rcvdTc for each and every MSTI if a TCN BPDU has been received, and extracts the message priority and timer values from the received BPDU storing them in the msgPriority and msgTimes variables.

The procedure sets rcvdMsg for the CIST, and makes the received CST or CIST message available to the CIST Port Information state machines.

If and only if rcvdInternal is set, this procedure sets rcvdMsg for each and every MSTI for which a MSTI message is conveyed in the BPDU, and makes available each MSTI message and the common parts of the CIST message priority (the CIST Root Identifier, External Root Path Cost, and Regional Root Identifier) to the Port Information state machine for that MSTI.

13.27.14 recordAgreement()

For the CIST and a given port, if rstpVersion is TRUE, operPointToPointMAC (6.6.3) is TRUE, and the received CIST Message has the Agreement flag set, then the CIST agreed flag is set and the CIST proposing flag is cleared. Otherwise the CIST agreed flag is cleared. Additionally, if the CIST message was received from a bridge in a different MST Region, i.e., the rcvdInternal flag is clear, the agreed and proposing flags for this port for all MSTIs are set or cleared to the same value as the CIST agreed and proposing flags. If the CIST message was received from a bridge in the same MST Region, the MSTI agreed and proposing flags are not changed.

For a given MSTI and port, if operPointToPointMAC (6.6.3) is TRUE, and

- a) The message priority vector of the CIST Message accompanying the received MSTI Message (i.e., received in the same BPDU) has the same CIST Root Identifier, CIST External Root Path Cost, and Regional Root Identifier as the CIST port priority vector, and
- b) The received MSTI Message has the Agreement flag set,

the MSTI agreed flag is set and the MSTI proposing flag is cleared. Otherwise the MSTI agreed flag is cleared.

NOTE—MSTI Messages received from bridges external to the MST Region are discarded and not processed by recordAgreement() or recordProposal().

13.27.15 recordDispute()

For the CIST and a given port, if the CIST message has the learning flag set:

- a) The disputed variable is set; and
- b) The agreed variable is cleared.

Additionally, if the CIST message was received from a bridge in a different MST region (i.e., if the rcvdInternal flag is clear), then for all the MSTIs:

- c) The disputed variable is set; and
- d) The agreed variable is cleared.

For a given MSTI and port, if the received MSTI message has the learning flag set:

- e) The disputed variable is set; and
- f) The agreed variable is cleared.

13.27.16 recordMastered()

For the CIST and a given port, if the CIST message was received from a bridge in a different MST Region, i.e. the rcvInternal flag is clear, the mastered variable for this port is cleared for all MSTIs.

For a given MSTI and port, if the MSTI message was received on a point-to-point link and the MSTI Message has the Master flag set, set the mastered variable for this MSTI. Otherwise reset the mastered variable.

13.27.17 recordPriority()

Sets the components of the portPriority variable to the values of the corresponding msgPriority components.

13.27.18 recordProposal()

For the CIST and a given port, if the received CIST Message conveys a Designated Port Role, and has the Proposal flag set, the CIST proposed flag is set. Otherwise the CIST proposed flag is not changed. Additionally, if the CIST Message was received from a bridge in a different MST Region, i.e., the rcvInternal flag is clear, the proposed flags for this port for all MSTIs are set or cleared to the same value as the CIST proposed flag. If the CIST message was received from a bridge in the same MST Region, the MSTI proposed flags are not changed.

For a given MSTI and port, if the received MSTI Message conveys a Designated Port Role, and has the Proposal flag set, the MSTI proposed flag is set. Otherwise the MSTI proposed flag is not changed.

13.27.19 recordTimes()

For the CIST and a given port, sets portTimes' Message Age, Max Age, Forward Delay, and remainingHops to the received values held in msgTimes and portTimes' Hello Time to the default specified in Table 13-5.

For a given MSTI and port, sets portTime's remainingHops to the received value held in msgTimes.

13.27.20 setReRootTree()

Sets reRoot TRUE for this tree (the CIST or a given MSTI) for all ports of the bridge.

13.27.21 setSelectedTree()

Sets selected TRUE for this tree (the CIST or a given MSTI) for all ports of the bridge if reselect is FALSE for all ports in this tree. If reselect is TRUE for any port in this tree, this procedure takes no action.

13.27.22 setSyncTree()

Sets sync TRUE for this tree (the CIST or a given MSTI) for all ports of the bridge.

13.27.23 setTcFlags()

For the CIST and a given port:

- a) If the Topology Change Acknowledgment flag is set for the CIST in the received BPDU, sets rcvdTcAck TRUE.
- b) If rcvInternal is clear and the Topology Change flag is set for the CIST in the received BPDU, sets rcvdTc TRUE for the CIST and for each and every MSTI.

- c) If rcvdInternal is set, sets rcvdTc for the CIST if the Topology Change flag is set for the CIST in the received BPDU.

For a given MSTI and port, sets rcvdTc for this MSTI if the Topology Change flag is set in the corresponding MSTI message.

13.27.24 setTcPropTree()

If and only if restrictedTcn is FALSE for the port that invoked the procedure, sets tcProp TRUE for the given tree (the CIST or a given MSTI) for all other ports.

13.27.25 syncMaster()

For all MSTIs, for each port that has infolInternal set:

- a) Clears the agree, agreed, and synced variables; and
- b) Sets the sync variable.

13.27.26 txConfig()

Transmits a Configuration BPDU. The first four components of the message priority vector (13.25.26) conveyed in the BPDU are set to the value of the CIST Root Identifier, External Root Path Cost, Bridge Identifier, and Port Identifier components of the CIST's designatedPriority parameter (13.25.7) for this port. The topology change flag is set if (tcWhile != 0) for the port. The topology change acknowledgment flag is set to the value of tcAck for the port. The remaining flags are set to zero. The value of the Message Age, Max Age, and Fwd Delay parameters conveyed in the BPDU are set to the values held in the CIST's designatedTimes parameter (13.25.8) for the port. The value of the Hello Time parameter conveyed in the BPDU is set to the value held in the CIST's portTimes parameter (13.25.34) for the port.

13.27.27 txRstp()

Transmits a RST BPDU or MST BPDU as determined by the value of ForceProtocolVersion (13.6.2), and encoded as specified by Clause 14. All per port variables referenced in this clause (13.27.27) are those for the transmitting port.

The first six components of the CIST message priority vector (13.25.26) conveyed in the BPDU are set to the value of the CIST's designatedPriority parameter (13.25.7). The Port Role in the BPDU (14.2.1) is set to the current value of the CIST's role (13.25.51). The Agreement and Proposal flags in the BPDU are set to the values of agree (13.25.3) and proposing (13.25.36), respectively. The CIST topology change flag is set if (tcWhile != 0) for the port. The topology change acknowledge flag in the BPDU is never used and is set to zero. The Learning and Forwarding flags in the BPDU are set to the values of learning (13.25.22) and forwarding (13.25.15) for the CIST, respectively. The value of the Message Age, Max Age, and Fwd Delay parameters conveyed in the BPDU are set to the values held in the CIST's designatedTimes parameter (13.25.8). The value of the Hello Time parameter conveyed in the BPDU is set to the value held in the CIST's portTimes parameter (13.25.34).

If the value of the Force Protocol Version parameter is less than 3, no further parameters are encoded in the BPDU and the protocol version parameter is set to 2 (denoting a RST BPDU). Otherwise, MST BPDU parameters are encoded:

- a) The version 3 length.
- b) The MST Configuration Identifier parameter of the BPDU is the value of MstConfigId (13.24.6).
- c) The CIST Internal Root Path Cost (13.25.7).
- d) The CIST Bridge Identifier (CIST Designated Bridge Identifier—13.25.7).

- e) The CIST Remaining Hops (13.25.8).
- f) The parameters of each MSTI message, encoded in MSTID order.

NOTE—No more than 64 MSTIs may be supported. The parameter sets for all of these can be encoded in a standard-sized Ethernet frame. The number of MSTIs supported can be zero.

If the value of the Force Protocol Version parameter is less than 3, no further parameters are encoded in the BPDU and the protocol version parameter is set to 3 (denoting a MST BPDU).

13.27.28 txTcn()

Transmits a TCN BPDU.

13.27.29 updtpdpuVersion()

Sets rcvdSTP TRUE if the BPDU received is a version 0 or version 1 TCN or a Config BPDU. Sets rcvdRSTP TRUE if the received BPDU is a RST BPDU or a MST BPDU.

13.27.30 updtrcvdInfoWhile()

Updates rcvdInfoWhile (13.23). The value assigned to rcvdInfoWhile is three times the Hello Time, if either:

- a) Message Age, incremented by 1 second and rounded to the nearest whole second, does not exceed Max Age and the information was received from a bridge external to the MST Region (rcvdInternal FALSE);

or

- b) remainingHops, decremented by one, is greater than zero and the information was received from a bridge internal to the MST Region (rcvdInternal TRUE);

and is zero otherwise.

The values of Message Age, Max Age, remainingHops, and Hello Time used in these calculations are taken from the CIST's portTimes parameter (13.25.34) and are not changed by this procedure.

13.27.31 updtrolesTree()

This procedure calculates the following priority vectors (13.8, 13.9 for the CIST, 13.10 for a MSTI) and timer values, for the CIST or a given MSTI:

- a) The *root path priority vector* for each Bridge Port that is not Disabled and has a *port priority vector* (portPriority plus portId—see 13.25.33 and 13.25.32) that has been recorded from a received message and not aged out (infols == Received).
- b) The Bridge's *root priority vector* (rootPortId, rootPriority—13.24.7, 13.24.8), chosen as the best of the set of priority vectors comprising the bridge's own *bridge priority vector* (BridgePriority—13.24.2) plus all calculated root path priority vectors whose:
 - 1) DesignatedBridgeID Bridge Address component is not equal to that component of the bridge's own bridge priority vector (13.9) and,
 - 2) Port's restrictedRole parameter is FALSE.
- c) The bridge's *root times*, (rootTimes—13.24.9), set equal to:
 - 1) BridgeTimes (13.24.3), if the chosen root priority vector is the bridge priority vector; otherwise,
 - 2) portTimes (13.25.34) for the port associated with the selected root priority vector, with the Message Age component incremented by 1 second and rounded to the nearest whole second if

the information was received from a bridge external to the MST Region (`rcvdInternal` FALSE), and with `remainingHops` decremented by one if the information was received from a bridge internal to the MST Region (`rcvdInternal` TRUE).

- d) The *designated priority vector* (`designatedPriority`—13.25.7) for each port; and
- e) The *designated times* (`designatedTimes`—13.25.8) for each port set equal to the value of *root times*.

If the root priority vector for the CIST is recalculated, and has a different Regional Root Identifier than that previously selected, and has or had a nonzero CIST External Root Path Cost, the `syncMaster()` procedure (13.27.25) is invoked.

NOTE—Changes in Regional Root Identifier will not cause loops if the Regional Root is within an MST Region, as is the case if and only if the MST Region is the Root of the CST. This important optimization allows the MSTIs to be fully independent of each other in the case where they compose the core of a network.

The CIST, or MSTI Port Role for each port is assigned, and its port priority vector and timer information are updated as specified in the remainder of this clause (13.39.2).

If the port is Disabled (`infols == Disabled`), `selectedRole` is set to `DisabledPort`.

Otherwise, if this procedure was invoked for an MSTI, for a port that is not Disabled, and that has CIST port priority information that was received from a bridge external to its bridge's Region (`infols == Received` and `infolInternal == FALSE`), then:

- f) If the selected CIST Port Role (calculated for the CIST prior to invoking this procedure for an MSTI) is `RootPort`, `selectedRole` is set to `MasterPort`.
- g) If selected CIST Port Role is `AlternatePort`, `selectedRole` is set to `AlternatePort`.
- h) Additionally, `updInfo` is set if the port priority vector differs from the designated priority vector or the port's associated timer parameter differs from the one for the Root Port.

Otherwise, for the CIST for a port that is not Disabled, or for an MSTI for a port that is not Disabled and whose CIST port priority information was not received from a bridge external to the Region (`infols != Received` or `infolInternal == TRUE`), the CIST or MSTI port role is assigned, and the port priority vector and timer information updated as follows:

- i) If the port priority vector information was aged (`infols = Aged`), `updInfo` is set and `selectedRole` is set to `DesignatedPort`;
- j) If the port priority vector was derived from another port on the bridge or from the bridge itself as the Root Bridge (`infols = Mine`), `selectedRole` is set to `DesignatedPort`. Additionally, `updInfo` is set if the port priority vector differs from the designated priority vector or the port's associated timer parameter(s) differ(s) from the Root Port's associated timer parameters;
- k) If the port priority vector was received in a Configuration Message and is not aged (`infols == Received`), and the root priority vector is now derived from it, `selectedRole` is set to `RootPort`, and `updInfo` is reset;
- l) If the port priority vector was received in a Configuration Message and is not aged (`infols == Received`), the root priority vector is not now derived from it, the designated priority vector is not better than the port priority vector, and the designated bridge and designated port components of the port priority vector do not reflect another port on this bridge, `selectedRole` is set to `AlternatePort`, and `updInfo` is reset;
- m) If the port priority vector was received in a Configuration Message and is not aged (`infols == Received`), the root priority vector is not now derived from it, the designated priority vector is not better than the port priority vector, and the designated bridge and designated port components of the port priority vector reflect another port on this bridge, `selectedRole` is set to `BackupPort`, and `updInfo` is reset;

- n) If the port priority vector was received in a Configuration Message and is not aged (`infols == Received`), the root priority vector is not now derived from it, the designated priority vector is better than the port priority vector, `selectedRole` is set to `DesignatedPort`, and `updInfo` is set.

13.27.32 `uptRolesDisabledTree()`

This procedure sets `selectedRole` to `DisabledPort` for all ports of the bridge for a given tree (CIST, or MSTI).

13.28 The Port Timers state machine

The Port Timers state machine shall implement the function specified by the state diagram in Figure 13-15 and the attendant definitions in 13.23 through 13.27.

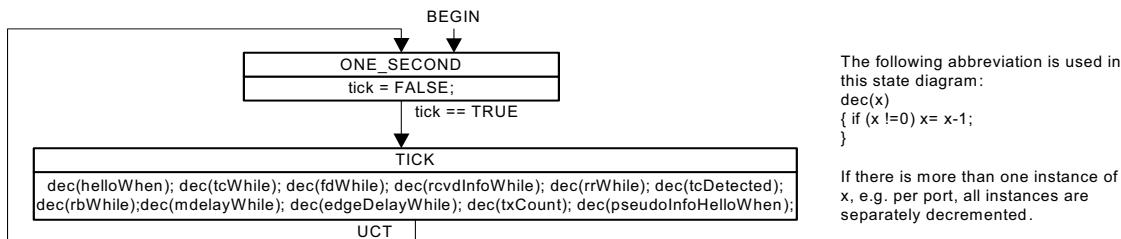


Figure 13-15—Port Timers state machine

The state machine uses `tick` (13.25), a signal set by an implementation-specific system clock function at one second intervals, to decrement the timer variables for the CIST and all MSTIs for the port. The state machine that uses a given timer variable is responsible for setting its initial value.

13.29 Port Receive state machine

The Port Receive state machine shall implement the function specified by the state diagram in Figure 13-16 and the attendant definitions in 13.23 through 13.27.

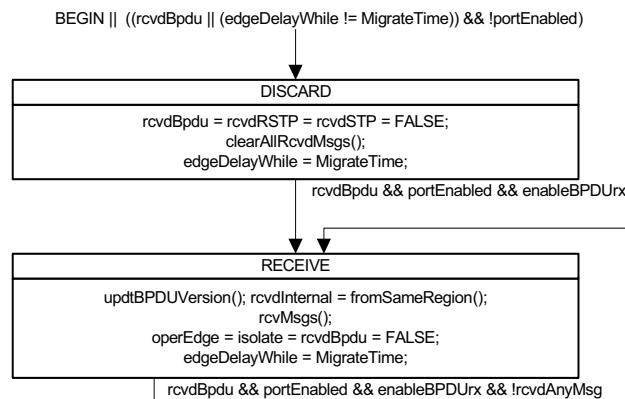


Figure 13-16—Port Receive state machine

This state machine receives and decodes validated BPDUUs. The `rcvdMsg` flag is set for the CIST, and for each MSTI supported by the receiving bridge if the received BPDU is from the same MST Region. The next BPDU is not processed until all `rcvdMsg` flags have been cleared by the per tree state machines.

NOTE—This standard does not specify or constrain the means used by the `recvMsgs()` procedure to identify or extract per tree information from a BPDU for processing by `rcvInfo()` in the Port Information Machine RECEIVE state.

13.30 Port Protocol Migration state machine

The Port Protocol Migration state machine shall implement the function specified by the state diagram in Figure 13-17 and the attendant definitions in 13.23 through 13.27.

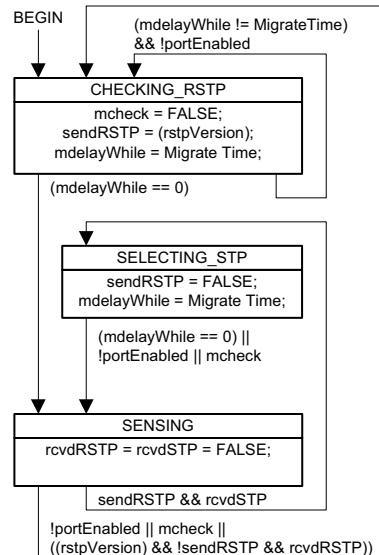


Figure 13-17—Port Protocol Migration state machine

13.31 Bridge Detection state machine

The Bridge Detection state machine shall implement the function specified by the state diagram in Figure 13-18 and the attendant definitions in 13.23 through 13.27.

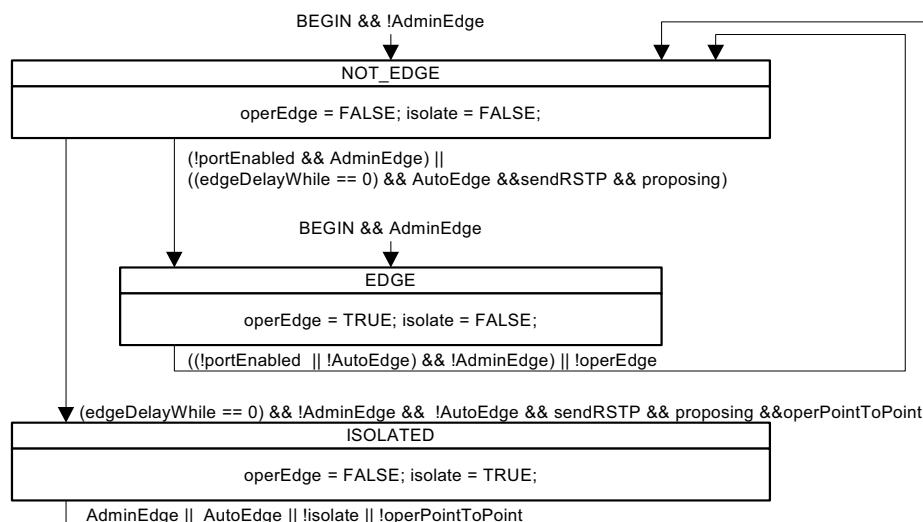
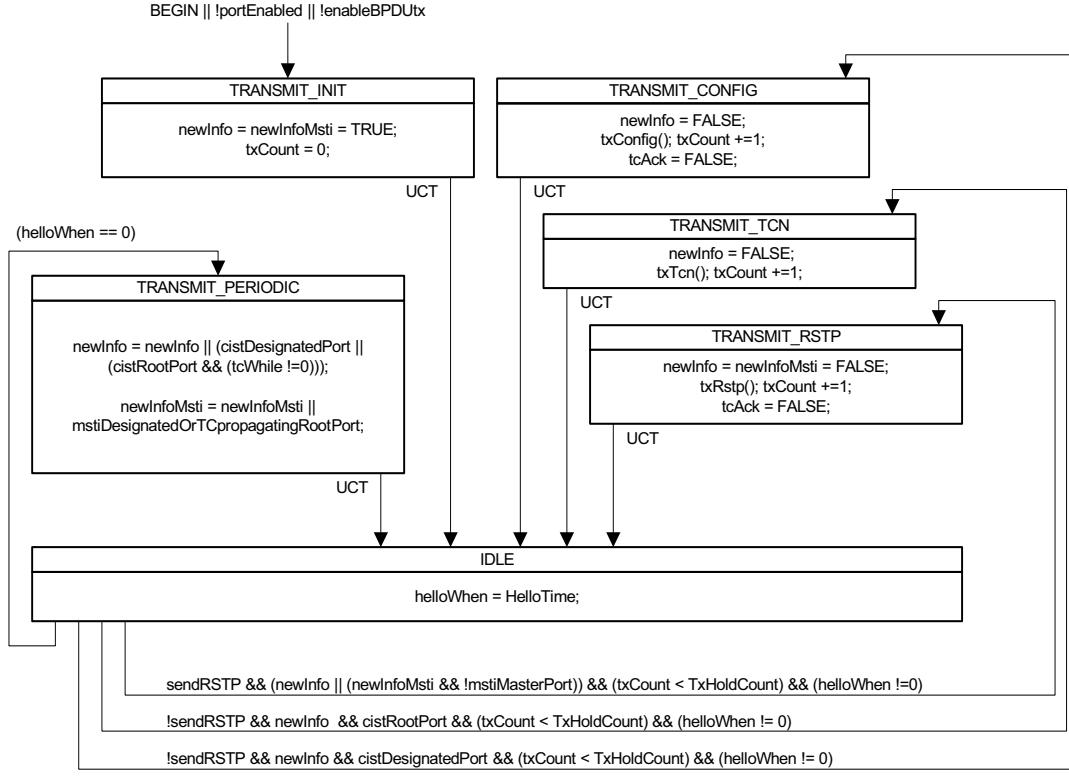


Figure 13-18—Bridge Detection state machine

13.32 Port Transmit state machine

The Port Transmit state machine shall implement the function specified by the state diagram in Figure 13-19 and the attendant definitions in 13.23 through 13.27.



All transitions, except UCT, are qualified by "&& allTransmitReady".

Figure 13-19—Port Transmit state machine

This state machine is responsible for transmitting BPDUs.

NOTE 1—Any single received BPDU that changes the CIST Root Identifier, CIST External Root Path Cost, or CIST Regional Root associated with MSTIs should be processed entirely, or not at all, before encoding BPDUs for transmission. This recommendation minimizes the number of BPDUs to be transmitted following receipt of a BPDU with new information. It is not required for correctness and has not therefore been incorporated into the state machines.

NOTE 2—If a CIST state machine sets **newInfo**, this machine will ensure that a BPDU is transmitted conveying the new CIST information. If MST BPDUs can be transmitted through the port, this BPDU will also convey new MSTI information for all MSTIs. If a MSTI state machine sets **newInfoMsti**, and MST BPDUs can be transmitted through the port, this machine will ensure that a BPDU is transmitted conveying information for the CIST and all MSTIs. Separate **newInfo** and **newInfoMsti** variables are provided to avoid requiring useless transmission of a BPDU through a port that can only transmit STP BPDUs (as required by the Force Protocol Version parameter or Port Protocol Migration machine) following a change in MSTI information without any change to the CIST.

13.33 Port Information state machine

The Port Information state machine for each tree shall implement the function specified by the state diagram in Figure 13-20 and the attendant definitions in 13.23 through 13.27.

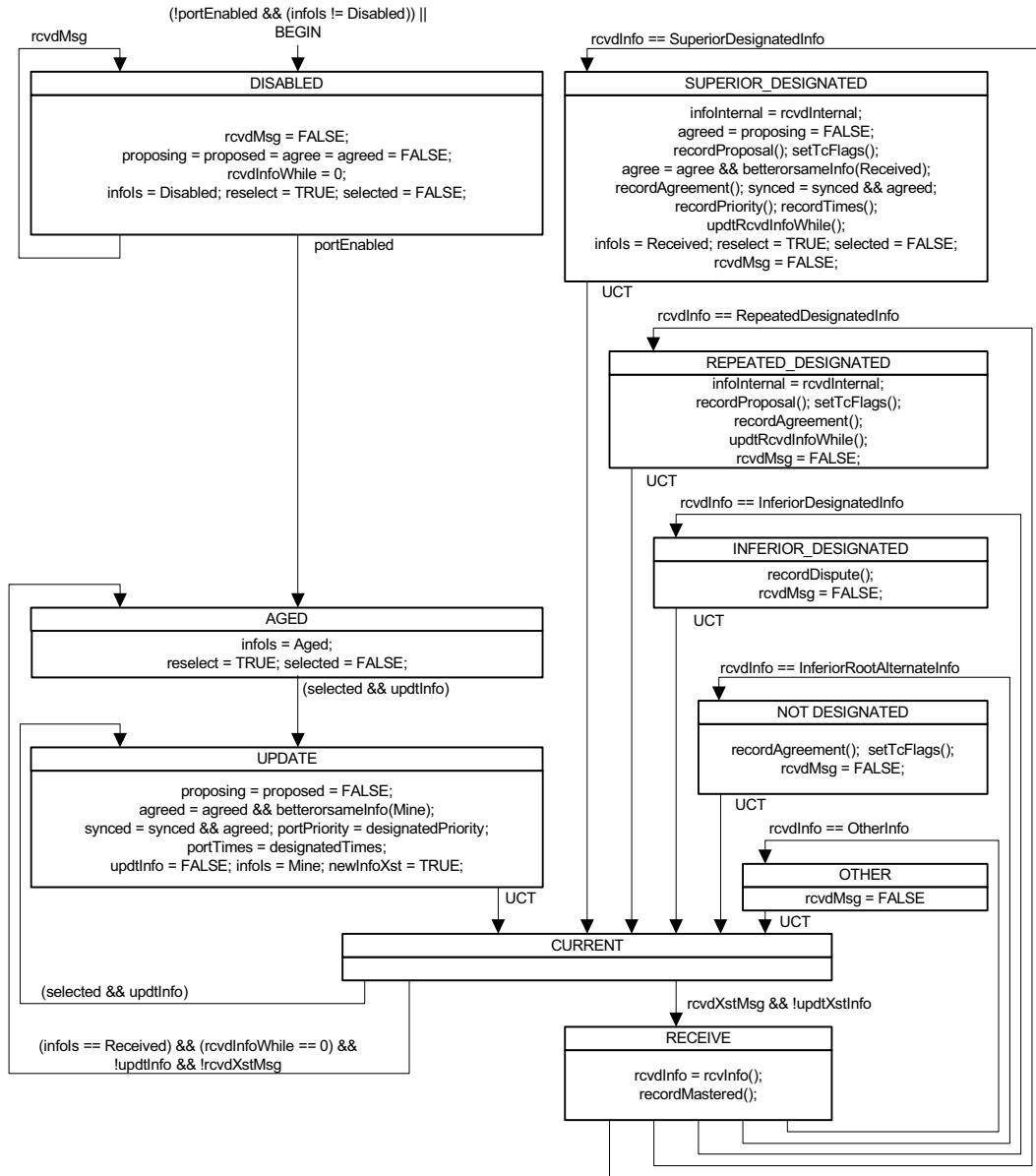


Figure 13-20—Port Information state machine

This state machine is responsible for recording the spanning tree information currently in use by the CIST or a given MSTI for a given port, ageing that information out if it was derived from an incoming BPDU, and recording the origin of the information in the infols variable. The selected variable is cleared and reselect set to signal to the Port Role Selection machine that port roles need to be recomputed. The infols and portPriority variables from all ports are used in that computation and, together with portTimes, determine new values of designatedPriority and designatedTimes. The selected variable is set by the Port Role Selection machine once the computation is complete.

13.34 Port Role Selection state machine

The Port Role Selection state machine shall implement the function specified by the state diagram in Figure 13-21 and the attendant definitions in 13.23 through 13.27.

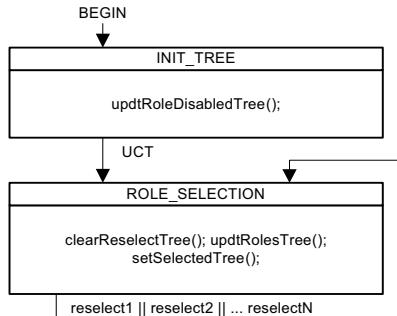


Figure 13-21—Port Role Selection state machine

13.35 Port Role Transitions state machine

The Port Role Transitions state machine shall implement the function specified by the state diagram in the following neighbors:

- Part 1: Figure 13-22 for both the initialization of this state machine and the states associated with the DisabledPort role; and
- Part 2: Figure 13-23 for the states associated with the MasterPort role; and
- Part 3: Figure 13-24 for the states associated with the RootPort role; and
- Part 4: Figure 13-25 for the states associated with the DesignatedPort role; and
- Part 5: Figure 13-26 for the states associated with the AlternatePort and BackupPort roles;

and the attendant definitions in 13.23 through 13.27.

As Figure 13-22, Figure 13-23, Figure 13-24, Figure 13-25, and Figure 13-26 are component parts of the same state machine, the global transitions associated with these diagrams are possible exit transitions from the states shown in any of the diagrams.

Figure 13-22 and Figure 13-26 show the Port Roles for ports that do not form part of the active topology of the given tree.

Figure 13-23, Figure 13-24, and Figure 13-25 show the Port Roles that form part of the active topology.

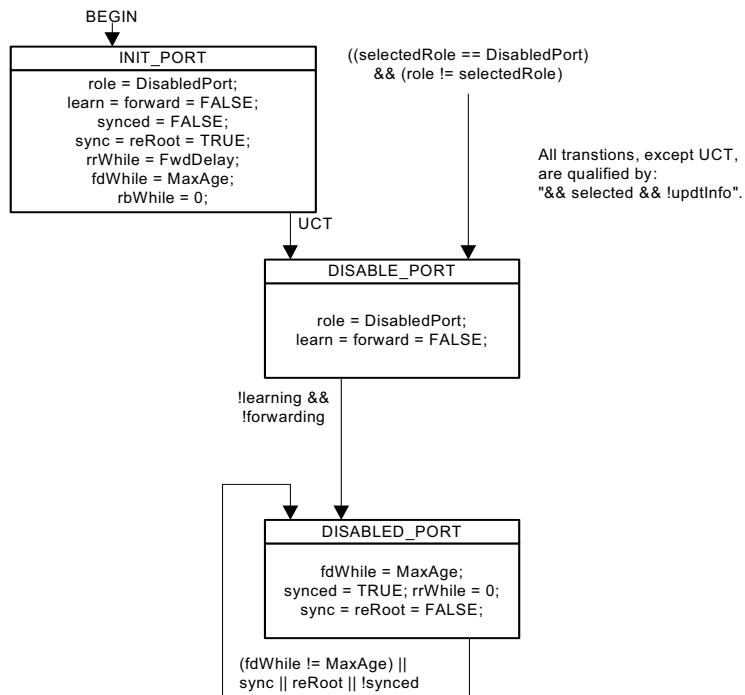


Figure 13-22—Disabled Port role transitions

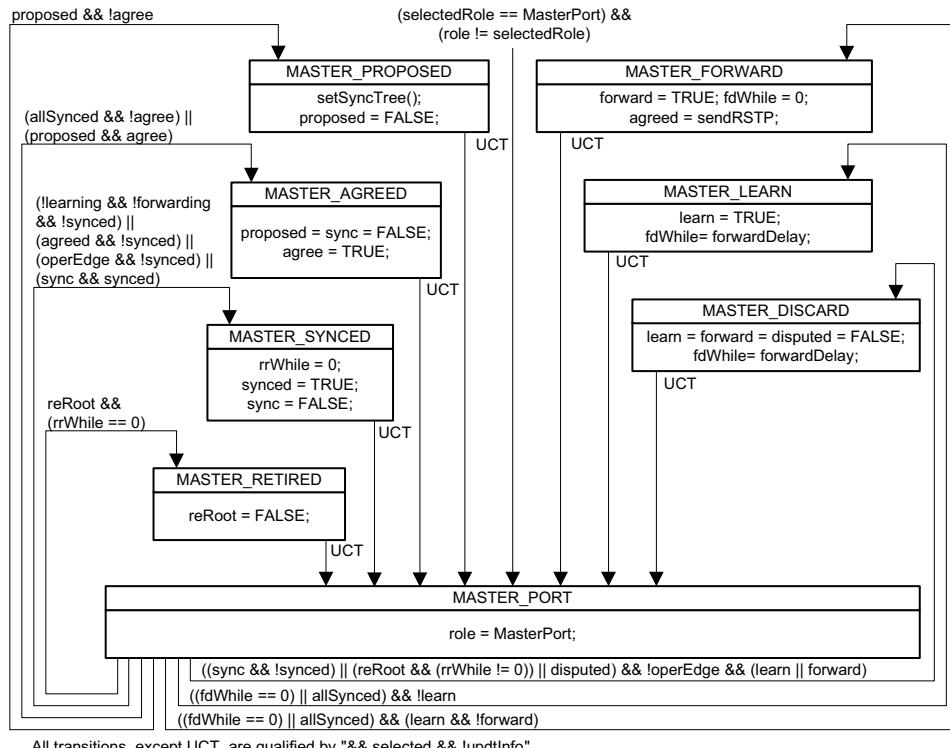
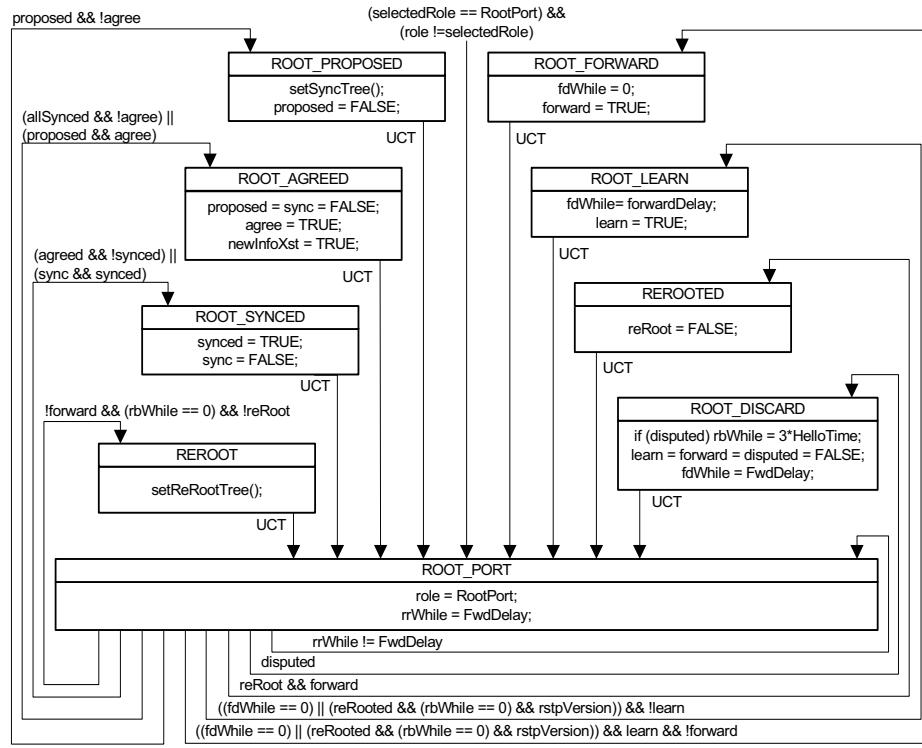


Figure 13-23—Port Role Transitions state machine—MasterPort



All transitions, except UCT, are qualified by "`&& selected && !updtnfo`".

Figure 13-24—Port Role Transitions state machine—RootPort

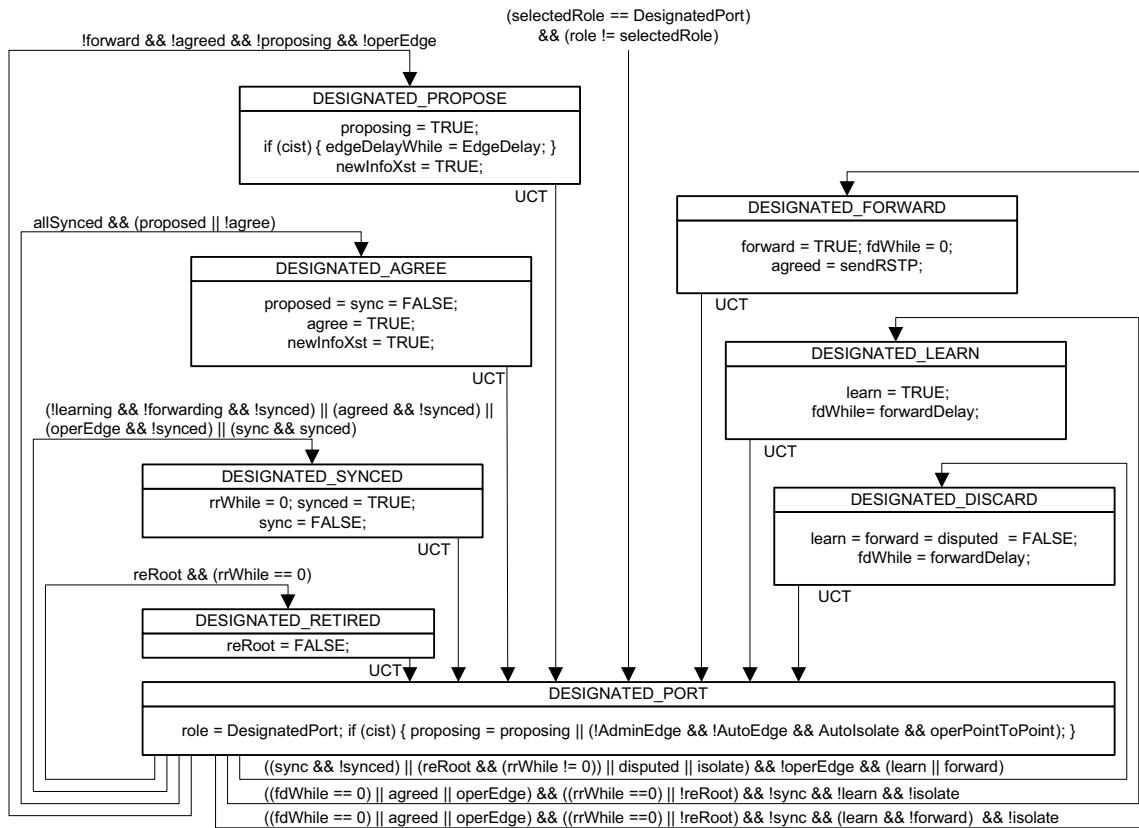


Figure 13-25—Port Role Transitions state machine—DesignatedPort

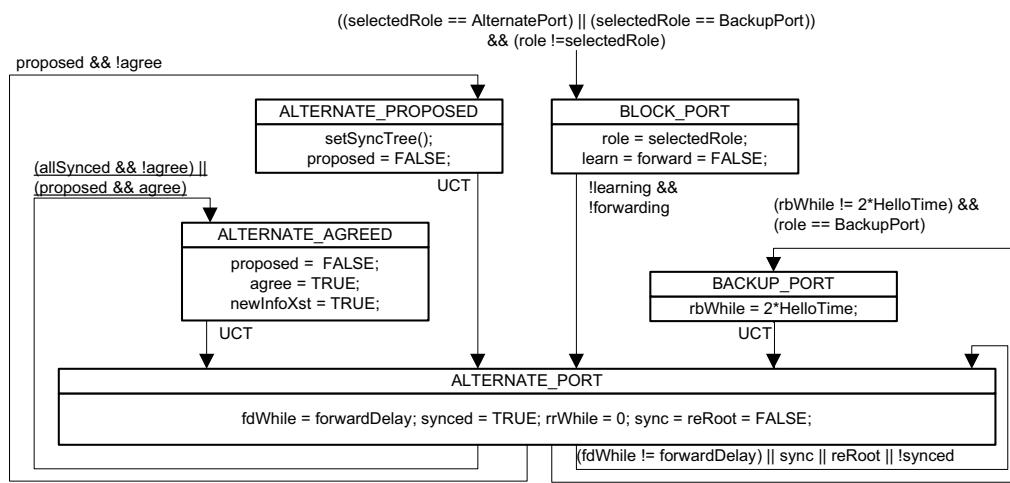


Figure 13-26—Port Role Transitions state machine—AlternatePort and BackupPort

13.36 Port State Transition state machine

The Port State Transition state machine shall implement the function specified by the state diagram in Figure 13-27 and the attendant definitions in 13.23 through 13.27.

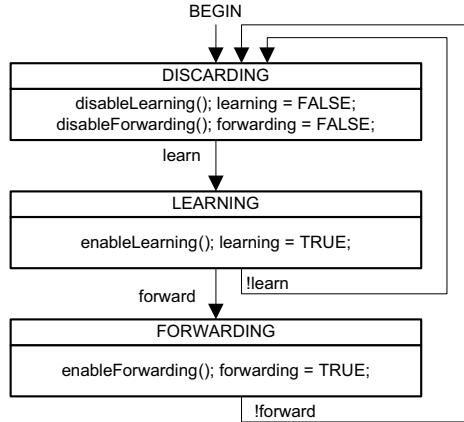


Figure 13-27—Port State Transition state machine

NOTE—A small system-dependent delay may occur on each of the transitions shown in the referenced state machine.

This state machine operates independently of the type of tree (CIST, or MSTI) and whether or not backbone bridging is being supported. However the way in which the bridge supports the learn and forward variables and the disableForwarding(), disableLearning(), enableForwarding(), and enableLearning() procedures are supported does vary (see 8.4, 8.6, 8.6.1). The forwarding and learning variables provide implementation independent reporting of the current state.

13.36.1 Port State transitions for the CIST and MSTIs

The CIST and each MSTI are always supported by an explicit Port State, enforced by bridge's implementation of the Forwarding Process, and the procedures prompt that implementation to take the necessary action to forward and/or learn from received frames (as requested).

13.37 Topology Change state machine

The Topology Change state machine for each tree shall implement the function specified by the state diagram in Figure 13-28 and the attendant definitions in 13.23 through 13.27.

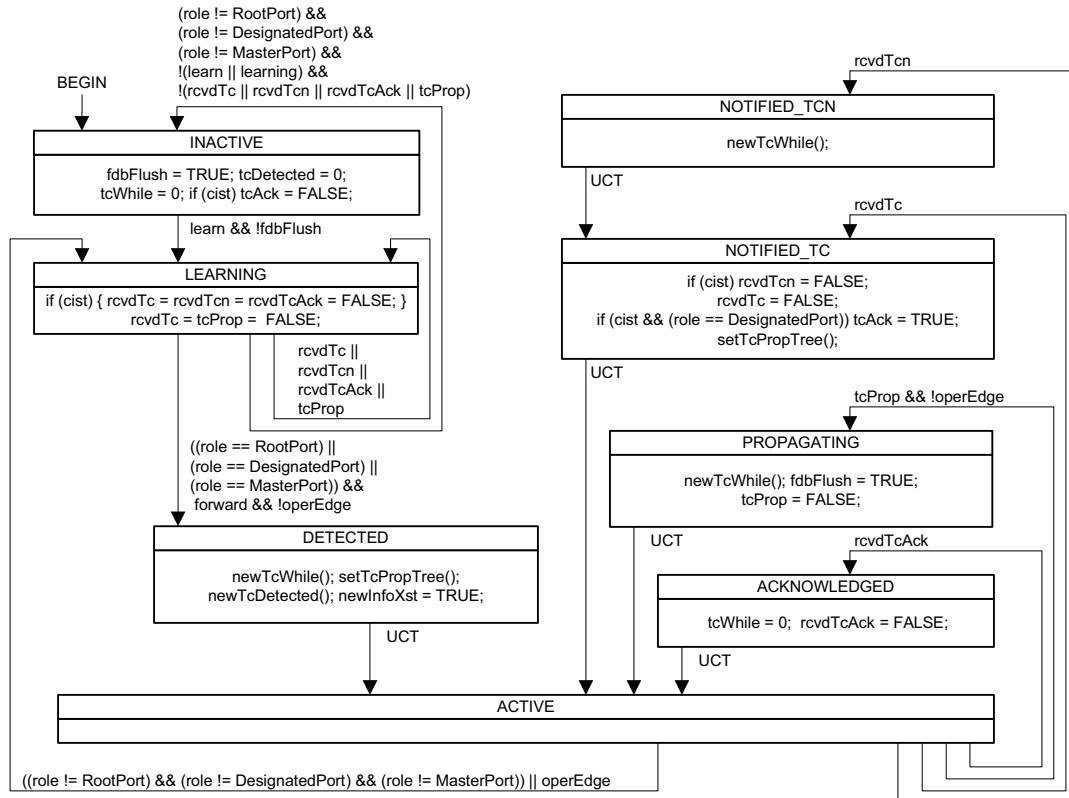


Figure 13-28—Topology Change state machine

NOTE—MRP (*Clause 10*) uses the `tcDetected` variable maintained by this state machine.

13.38 Layer 2 Gateway Port Receive state machine

If implemented, the Layer 2 Gateway Port state machine for each port shall implement the function specified by the state diagram in Figure 13-29 and the attendant definitions in 13.23 through 13.27.

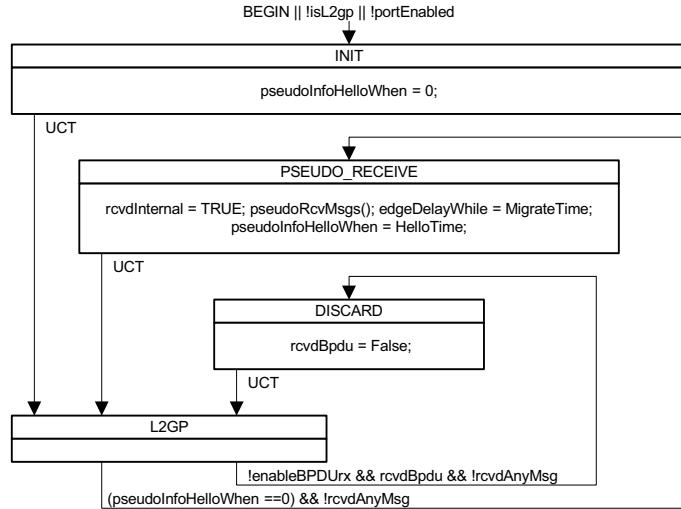


Figure 13-29—L2 Gateway Port Receive state machine

NOTE—The functionality provided by this state machine is discussed in 13.18, 25.9.2, 26.4.3, and 13.27.11.

13.39 Customer Edge Port Spanning Tree operation

This subclause specifies the operation of the Spanning Tree Protocol Entity within a C-VLAN component that supports a Customer Edge Port (Figure 15-4) of a Provider Edge Bridge. The Customer Edge Port and each Provider Edge Port are treated as separate Bridge Ports by the spanning tree protocol.

If the C-VLAN component connects to the S-VLAN component with a single Provider Edge Port, then all frames (including Spanning Tree BPDUs) addressed to the Bridge Group Address may be relayed between the two ports of the C-VLAN component without modification. Otherwise, the Spanning Tree Protocol Entity shall execute RSTP, as modified by the provisions of this subclause (13.39).

The RSTP enhancements specified do not reduce Provider Bridged Network connectivity between Customer Edge Ports to a single spanning tree of service instances but ensure that connectivity for frames assigned to any given C-VLAN is loop-free. In this respect, the C-VLAN component's spanning tree protocol operation is equivalent to, but simpler to manage than, the operation of MSTP.

13.39.1 Provider Edge Port `operPointToPointMAC` and `operEdge`

The value of the `adminPointToPointMAC` parameter for a Provider Edge Port is always Auto, and no management control over its setting is provided. The value of the `operPointToPointMAC` parameter, used by the RSTP state machines, shall be true if the service instance corresponding to the Provider Edge Port connects at most two customer interfaces, and false otherwise.

The value of the `AdminEdge`, `AutoEdge`, and `operEdge` parameters for a Provider Edge Port are always false, true, and false, respectively. No management control over their setting is provided.

13.39.2 **updRolesTree()**

The spanning tree priority vectors timer values are calculated, and the port role for each port, its port priority vector, and timer information updated as specified in 13.27.31, with one exception. If selectedRole was to be set to AlternatePort, the port is an Provider Edge Port, and the root priority vector was derived from another Provider Edge Port, then the selectedRole shall be set to Root Port.

NOTE—The effect of this enhancement is to allow the C-VLAN component to have multiple Root Ports (just as if separate per S-VLAN trees were being provided), if they are all Provider Edge Ports. As the C-VLAN component assigns each frame to a single C-VLAN and maps any given C-VLAN to and from at most one Provider Edge Port, no loop is created.

13.39.3 **setReRootTree(), setSyncTree(), setTcPropTree()**

The **setReRootTree()** and **setSyncTree()** procedures specified in 13.27.20 and 13.27.22 set the **reRoot** and **sync** variables for all ports of the bridge, and the **setTcPropTree()** as specified in 13.27.24 sets the **tcProp** variable for all ports other than the port that invoked the procedure. If the port invoking the procedure is a Customer Edge Port, then this behavior is unchanged; if it is a Provider Edge Port, then the behavior of each procedure shall be as follows.

The **setReRootTree()** procedure sets **reRoot** for the port invoking the procedure and for the Customer Edge Port.

The **setSyncTree()** procedure sets **sync** for the port invoking the procedure and for the Customer Edge Port.

The **setTcPropTree()** procedure sets **tcProp** for the Customer Edge Port.

13.39.4 **allSynced, reRooted**

RSTP specifies a single value of the **allSynced** and **reRooted** state machine conditions for all Bridge Ports. This specification requires an independent value of each of these conditions for each port of the C-VLAN component. If that port is the Customer Edge Port, then **allSynced** shall be true if and only if **synced** is true for all Provider Edge Ports, and **reRooted** shall be true if and only if **rrWhile** is zero for all Provider Edge Ports. If the port for which the condition is being evaluated is a Provider Edge Port, then **allSynced** shall take the value of **synced** for the Customer Edge Port, and **reRooted** shall be true if and only if **rrWhile** is zero for the Customer Edge Port.

13.39.5 Configuration parameters

All configuration parameters for RSTP should be set to their recommended defaults, with the exception of the following, which are chosen to minimize the chance of interfering with the customer's configuration (e.g., by the C-VLAN component becoming the root of the customer spanning tree), as follows:

- a) The Bridge Priority (13.16, Table 13-3, 13.24.2) should be set to 61 440. This sets the priority part of the Bridge Identifier (the most significant 4 bits) to hex F.
- b) The following 12 bits (the Bridge Identifier system ID extension) should be set to hex FFF.
- c) The Port Priority (13.16, Table 13-3, 13.25.33) should be set to 32. This sets the priority part of the Port Identifier (the most significant 4 bits) to hex 2, a higher priority than the default (128, or hex 8).
- d) The Port Path Cost values for Provider Edge Ports should be set to 128.

All BPDUs generated by the Spanning Tree Protocol Entity within a C-VLAN component use the MAC address of the Customer Edge Port as a source address and as the bridge address portion of the Bridge Identifier. For each internal Provider Edge Port, the protocol uses the S-VID associated with the

corresponding internal Customer Network Port on the S-VLAN component as a port number. For the Customer Edge Port, the value 0xFFFF is used as the port number.

13.40 Virtual Instance Port Spanning Tree operation

This subclause specifies the operation of the Spanning Tree Protocol Entity within an I-component in a Backbone Edge Bridge. The Customer Network Ports (CNP) and Virtual Instance Ports (VIP) are treated as separate Bridge Ports by the spanning tree protocol.

If the I-component has a single CNP and a single VIP supported by a point-to-point backbone service instance, then all frames (including Spanning Tree BPDUs) addressed to the Provider Bridge Group address may be relayed between the two ports of the I-component without modification. Otherwise, the Spanning Tree Protocol Entity shall execute RSTP, as modified by the provisions of this subclause.

The RSTP enhancements specified ensure that connectivity for frames assigned to any given S-VLAN is loop-free.

The parameters and functions of RSTP used on the VIPs get the same values and functionality as defined for Provider Edge Ports of a C-VLAN component as defined in 13.39. The Bridge Identifier Priority and system ID extension get the values specified in 13.39.5. These changes in RSTP ensure that no VIP is blocked due to the operation of RSTP and the I-component will never be elected as root.

NOTE—The effect of not blocking any VIP in the I-component (never set the port role alternate to a VIP) will not cause a loop since the I-component maps any given S-VID to at most one VIP.

14. Use of BPDU by MSTP

This clause specifies the BPDU formats, encoding, and decoding used to exchange protocol parameters with other Bridges operating MSTP, RSTP, or STP, by a Bridge Protocol Entity operating MSTP (Clause 13).

14.1 BPDU Structure

14.1.1 Transmission and representation of octets

All BPDU shall contain an integral number of octets. The octets in a BPDU are numbered starting from 1 and increasing in the order they are put into a Data Link Service Data Unit (DLSDU). When bit positions in an octet or a sequence of octets encode a number, the number is encoded as an unsigned binary numeral with bit positions in lower octet numbers having more significance. Within an octet, the bits are numbered from 8 to 1, where 1 is the low-order bit. Where sequences of bits are represented, higher order bits are shown to the left of lower order bits in the same octet, and bits in lower octet numbers are shown to the left of bits in higher octet numbers.

14.1.2 Components

A Protocol Identifier is encoded in the initial octets of all BPDU. The single Protocol Identifier value of 0000 0000 0000 0000 identifies the Spanning Tree family of protocols (the Spanning Tree Algorithm and Protocol, the Rapid Spanning Tree Algorithm and Protocol, and the Multiple Spanning Tree Protocol).

14.2 Encoding of parameter types

The following parameter types are encoded as specified in 9.2 of IEEE Std 802.1D-2004:

- a) Protocol Identifiers
- b) Protocol Version Identifiers
- c) BPDU Types
- d) Flags
- e) Port Identifiers
- f) Timer values
- g) Length values

Additional considerations follow for encoding:

- h) Port Roles
- i) Bridge Identifiers
- j) Port Identifiers
- k) External Root Path Costs
- l) Internal Root Path Costs

This standard specifies new or extended parameter types and encodings for

- m) Hop Counts

14.2.1 Encoding of Port Role values

Port Role values shall be encoded in two consecutive flag bits, taken to represent an unsigned integer, as follows:

- a) A value of 0 indicates Master Port;
- b) A value of 1 indicates Alternate or Backup;
- c) A value of 2 indicates Root;
- d) A value of 3 indicates Designated.

14.2.2 Allocation and encoding of Bridge Identifiers

The 12-bit system ID extension component of a Bridge Identifier (9.2.5 of IEEE Std 802.1D-2004) is used to allocate distinct Bridge Identifiers to each Spanning Tree instance supported by the operation of MSTP, based on the use of a single Bridge Address component value for the MST Bridge as a whole. The system ID extension value zero shall be allocated to the Bridge Identifier used by MSTP in support of the CIST; the system ID extension value allocated to the Bridge Identifier used by a given MSTI shall be equal to the MSTID.

NOTE 1—This convention is used to convey the MSTID for each MSTI parameter set in an MST BPDU.

The 4 most significant bits of the Bridge Identifier for a given Spanning Tree instance (the settable Priority component) can be modified independently of the other Bridge Identifiers supported by the Bridge, allowing full configuration control to be exerted over each Spanning Tree instance with regard to bridge priority.

NOTE 2—Only these four bits of the transmitting Bridge’s Bridge Identifier are encoded in BPDUs for each MSTI. The remainder of the Bridge Identifier is derived from the CIST Bridge Identifier and the MSTID using the system ID extension convention described previously.

14.2.3 Allocation and encoding of Port Identifiers

The 4 most significant bits of the Port Identifier for a given Spanning Tree instance (the settable Priority component) can be modified independently for each Spanning Tree instance supported by the Bridge.

NOTE—Only these four bits of the transmitting Bridge’s Port Identifier are encoded in a BPDU for each MSTI. The remainder of the Port Identifier is derived from the CIST Port Identifier.

14.2.4 Encoding of External Root Path Cost

The External Root Path Cost shall be encoded as specified by 9.2.6 of IEEE Std 802.1D-2004 for Root Path Cost in four octets, taken to represent a number of arbitrary cost units. Subclause 17.4 of IEEE Std 802.1D-2004 contains recommendations as to the increment to the Root Path Cost, in order that some common value can be placed on this parameter without requiring a management installation practice for Bridges in a network.

14.2.5 Encoding of Internal Root Path Cost

The Internal Root Path Cost shall be encoded in four octets, taken to represent a number of arbitrary cost units that may differ from those used for External Path Cost. Table 13-4 contains recommendations for the use of these units. These recommendations allow higher LAN speeds to be represented in support of both current and future technologies, while still allowing common values to be assigned without a management installation practice.

NOTE—This revision from the original IEEE 802.1D recommendations for STP Path Cost causes no operational difficulties because there was no installed base of Bridges using the Internal Root Path Cost parameter prior to approval of this standard.

14.2.6 Encoding of Hop Counts

The number of remaining Hops parameter shall be encoded in a single octet.

14.3 BPDU formats and parameters

14.3.1 STP BPDUs

The formats of STP BPDU Configuration and TCN BPDUs are as specified in Clause 9 of IEEE Std 802.1D-2004.

14.3.2 RST BPDUs

The format of RST BPDUs is as specified in Clause 9 of IEEE Std 802.1D-2004.

14.3.3 MST BPDUs

The format of MST BPDUs is compatible with that specified for RST BPDUs (Clause 9 of IEEE Std 802.1D-2004), with the addition of fields to convey information for the IST and each MSTI and is shown in Figure 14-1. Each transmitted MST BPDU shall contain the parameters specified and no others.

NOTE—The BPDU specified in this clause is carried in an LLC Type 1 frame following the DSAP, LSAP, and UI fields (7.12 on Addressing in IEEE Std 802.1D-2004). The consequence of the inclusion of those three octets in an IEEE 802.3 or Ethernet MAC frame is that, if the MAC Addresses in the frame are aligned on an even octet boundary, then so are the BPDU octet pairs 6 and 7, 14 and 15, 18 and 19, etc.

	Octet
Protocol Identifier	1–2
Protocol Version Identifier	3
BPDU Type	4
CIST Flags	5
CIST Root Identifier	6–13
CIST External Path Cost	14–17
CIST Regional Root Identifier	18–25
CIST Port Identifier	26–27
Message Age	28–29
Max Age	30–31
Hello Time	32–33
Forward Delay	34–35
Version 1 Length = 0	36
Version 3 Length	37–38
MST Configuration Identifier	39–89
CIST Internal Root Path Cost	90–93
CIST Bridge Identifier	94–101
CIST Remaining Hops	102
MSTI Configuration Messages (may be absent)	103–39 + <i>Version 3 Length</i>

Figure 14-1—MST BPDU parameters and format

14.4 Validation of received BPDUs

An MST Bridge Protocol Entity shall examine Octets 1 and 2 (conveying the Protocol Identifier), Octet 3 (conveying the Protocol Version Identifier encoded as a number), Octet 4 (conveying the BPDU Type), and

the total length of the received BPDU (including the preceding fields, but none prior to the Protocol Identifier) to determine the further processing required as follows:

- a) If the Protocol Identifier is 0000 0000 0000 0000, the BPDU Type is 0000 0000, and the BPDU contains 35 or more octets, it shall be decoded as an STP Configuration BPDU.
- b) If the Protocol Identifier is 0000 0000 0000 0000, the BPDU Type is 1000 0000 (where bit 8 is shown at the left of the sequence), and the BPDU contains 4 or more octets, it shall be decoded as an STP TCN BPDU (9.3.2 of IEEE Std 802.1D-2004).
- c) If the Protocol Identifier is 0000 0000 0000 0000, the Protocol Version Identifier is 2, the BPDU Type is 0000 0010 (where bit 8 is shown at the left of the sequence), and the BPDU contains 36 or more octets, it shall be decoded as an RST BPDU.
- d) If the Protocol Identifier is 0000 0000 0000 0000, the Protocol Version Identifier is 3 or greater, the BPDU Type is 0000 0010, and the BPDU:
 - 1) Contains 35 or more but less than 103 octets; or
 - 2) Contains a Version 1 Length that is not 0; or
 - 3) Contains a Version 3 length that does not represent an integral number, from 0 to 64 inclusive, of MSTI Configuration Messages;
 it shall be decoded as an RST BPDU.
- e) If the Protocol Identifier is 0000 0000 0000 0000, the Protocol Version Identifier is 3 or greater, the BPDU Type is 0000 0010, and the BPDU contains:
 - 1) 102 or more octets; and
 - 2) A Version 1 Length of 0; and
 - 3) A Version 3 length representing an integral number, from 0 to 64 inclusive, of MSTI Configuration Messages;
 it shall be decoded as an MST BPDU.
- f) Otherwise the BPDU shall be discarded and not processed.

NOTE 1—The LLC LSAP that identifies BPDUs is reserved for standard protocols; no other protocols using that LSAP have been standardized although they may be at some future time. At that time, BPDUs with different Protocol Identifiers may be processed according to the rules of those protocols but will still be discarded from the point of view of MSTP.

NOTE 2—These validation rules are in accord with the approach to backward compatibility of future version enhancements set out in 9.3.4 of IEEE Std 802.1D-2004. Test a) and test b) do not check the Protocol Version Identifier.

NOTE 3—These validation rules do not contain a loopback check of the form specified in 9.3.4 of IEEE Std 802.1D-2004.

14.5 Transmission of BPDUs

An MST Bridge Protocol Entity shall encode 0000 0000 0000 0000 in Octets 1 and 2 (conveying the Protocol Identifier); the remaining fields shall be encoded to convey an STP Configuration BPDU, an STP TCN BPDU, an RST BPDU, or an MST BDU as required by the Force Protocol Version parameter, the Port Protocol Migration state machine, and other protocol parameters, all as specified in Clause 13.

- a) If transmission of an STP Configuration BPDU is required, the Protocol Version Identifier shall be 0, and the BPDU Type shall be 0000 0000.
- b) If transmission of an STP TCN BPDU is required, the Protocol Version Identifier shall be 0, and the BPDU Type shall be 1000 0000.
- c) If transmission of an RST BPDU is required, the Protocol Version Identifier shall be 2, and the BPDU Type shall be 0000 0010.
- d) If transmission of an MST BPDU is required, the Protocol Version Identifier shall be 3, and the BPDU Type shall be 0000 0010.

The remaining parameters for STP Configuration, RST, and MST BPDUs shall be encoded as follows.

14.6 Encoding and decoding of STP Configuration, RST, and MST BPDUs

STP Configuration, RST, and MST BPDU protocol parameters are encoded for transmission, and decoded, checked, or ignored on receipt as follows:

- a) Bit 1 of Octet 5 conveys the CIST Topology Change flag.
- b) Bit 2 of Octet 5 conveys the CIST Proposal flag in RST and MST BPDUs. It is unused in STP Configuration BPDUs and shall be transmitted as 0 and ignored on receipt.
- c) Bits 3 and 4 of Octet 5 conveys the CIST Port Role in RST and MST BPDUs. It is unused in STP Configuration BPDUs and shall be transmitted as 0 and ignored on receipt.
- d) Bit 5 of Octet 5 conveys the CIST Learning flag in RST and MST BPDUs. It is unused in STP Configuration BPDUs and shall be transmitted as 0 and ignored on receipt.
- e) Bit 6 of Octet 5 conveys the CIST Forwarding flag in RST and MST BPDUs. It is unused in STP Configuration BPDUs and shall be transmitted as 0 and ignored on receipt.
- f) Bit 7 of Octet 5 conveys the CIST Agreement flag in RST and MST BPDUs. It is unused in STP Configuration BPDUs and shall be transmitted as 0 and ignored on receipt.
- g) Bit 8 of Octet 5 conveys the Topology Change Acknowledge Flag in STP Configuration BPDUs. It is unused in RST and MST BPDUs and shall be transmitted as 0 and ignored on receipt.
- h) Octets 6 through 13 convey the CIST Root Identifier.

NOTE 1—The 12-bit system id extension component of the CIST Root Identifier can be received and subsequently transmitted as an arbitrary value, even in MST BPDUs, since the CIST Root may be an STP Bridge.

- i) Octets 14 through 17 convey the CIST External Root Path Cost.
- j) Octets 18 through 25 shall take the value of the CIST Regional Root Identifier when transmitted in RST and MST BPDUs, and the value of the CIST Bridge Identifier of the transmitting Bridge when transmitted in STP Configuration BPDUs. On receipt of an STP Configuration or RST BPDU, both the CIST Regional Root Identifier and the CIST Designated Bridge Identifier shall be decoded from this field. On receipt of an MST BPDU, the CIST Regional Root Identifier shall be decoded from this field.
- k) Octets 26 and 27 convey the CIST Port Identifier of the transmitting Bridge Port.
- l) Octets 28 and 29 convey the Message Age timer value.
- m) Octets 30 and 31 convey the Max Age timer value.
- n) Octets 32 and 33 convey the Hello Time timer value used by the transmitting Bridge Port.
- o) Octets 34 and 35 convey the Max Age timer value.

No further octets shall be encoded in STP Configuration BPDUs. Additional octets in received BPDUs identified by the validation procedure (14.4) as STP Configuration BPDUs shall be ignored. The specification of encoding or decoding of further octets in this subclause refers only to RST and MST BPDUs.

- p) Octet 36 conveys the Version 1 Length. This shall be transmitted as 0. It is checked on receipt by the validation procedure (14.4).

No further octets shall be encoded in RST BPDUs. Additional octets in received BPDUs identified by the validation procedure (14.4) as RST BPDUs shall be ignored. The specification of encoding or decoding of further octets in this subclause refers only to MST BPDUs.

NOTE 2—As Version 2 does not specify any additional fields beyond the end of the Version 0 information, there is no Version 2 Length field specified in Version 2 of the protocol (see Clause 9 of IEEE Std 802.1D-2004) and, therefore, no need for a Version 2 length field here.

- q) Octets 37 and 38 convey the Version 3 Length. Its value is the number of octets taken by the parameters that follow in the BPDU. It is checked on receipt by the validation procedure (14.4).
- r) Octets 39 through 89 convey the elements of the MST Configuration Identifier (13.7):
 - 1) The Configuration Identifier Format Selector is encoded in Octet 39 and shall take the value 0000 0000;
 - 2) The Configuration Name is encoded in octets 40 through 71;
 - 3) The Revision Level is encoded as a number in octets 72 through 73;
 - 4) The Configuration Digest is encoded in octets 74 through 89.
- s) Octets 90 through 93 convey the CIST Internal Root Path Cost.
- t) Octets 94 through 101 convey the CIST Bridge Identifier of the transmitting Bridge. The 12-bit system id extension component of the CIST Bridge Identifier shall be transmitted as 0. The behavior on receipt is unspecified if it is nonzero.

NOTE 3—The 4 most significant bits of the Bridge Identifier constitute the manageable priority component for each MSTI and are separately encoded in MSTI Configuration Messages in the BPDU.

NOTE 4—The 4 most significant bits constitute the manageable priority component of each MSTI and are separately encoded in MSTI Configuration Messages in the BPDU.

- u) Octet 102 encodes the value of remaining Hops for the CIST.
- v) A sequence of zero or more, up to a maximum of 64, MSTI Configuration Messages follows, each encoded as specified in 14.6.1.

14.6.1 MSTI Configuration Messages

A single instance of the following set of parameters is encoded for each MSTI supported by the transmitting Bridge.

- a) Bits 1, 2, 3 and 4, 5, 6, 7, and 8, respectively, of Octet 1 convey the Topology Change flag, Proposal flag, Port Role, Learning flag, Forwarding flag, Agreement flag, and Master flag for this MSTI.
- b) Octets 2 through 9 convey the Regional Root Identifier (13.25.7) as illustrated in Figure 14-2. This includes the value of the MSTID for this Configuration Message encoded in bits 4 through 1 of Octet 1, and bits 8 through 1 of Octet 2.

NOTE—The 4 most significant bits of each MSTI's Regional Root Identifier constitute a manageable priority component.

- c) Octets 10 through 13 convey the Internal Root Path Cost.
- d) Bits 5 through 8 of Octet 14 convey the value of the Bridge Identifier Priority for this MSTI. Bits 1 through 4 of Octet 14 shall be transmitted as 0, and ignored on receipt.
- e) Bits 5 through 8 of Octet 15 convey the value of the Port Identifier Priority for this MSTI. Bits 1 through 4 of Octet 15 shall be transmitted as 0, and ignored on receipt.
- f) Octet 16 conveys the value of remainingHops for this MSTI (13.25.8).

Octet
MSTI Flags
1
MSTI Regional Root Identifier
2–9
MSTI Internal Root Path Cost
10–13
MSTI Bridge Priority
14
MSTI Port Priority
15
MSTI Remaining Hops
16

Figure 14-2—MSTI Configuration Message parameters and format

15. Support of the MAC Service by Provider Bridged Networks

Provider Bridges interconnect the separate MACs of the IEEE 802 LANs that compose a Provider Bridged Network, relaying frames to provide connectivity between all LANs that provide customer interfaces for each service instance. The position of the Provider Bridge S-VLAN component bridging function within the MAC Sublayer is shown in Figure 15-1.

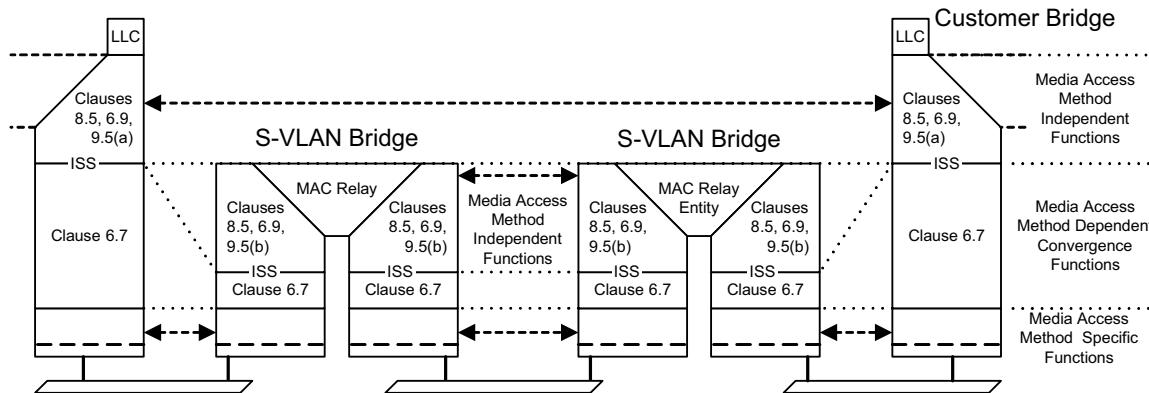


Figure 15-1—Internal organization of the MAC sublayer in a Provider Bridged Network

This clause discusses the following aspects of provisioning service instances on a Provider Bridged Network:

- a) Service transparency
- b) Customer service interfaces
- c) Service instance segregation
- d) Service instance selection and identification
- e) Service priority selection
- f) Service access protection

NOTE—In describing the MAC Service, this standard makes use of term “service” as defined by the OSI Reference Model (ISO 7498-1). In this sense, a service comprises a set of primitives and associated parameters, provided by one protocol layer in the architectural model to the protocol layer above, and the causal relationships between the primitives invoked by an upper layer protocol entity in one system with those resulting indications to a peer entity in another system. The term “service” used by service providers, while including layering concepts, goes far beyond this formal definition, and commonly specifies some or all of the following: interfacing considerations across multiple protocol layers (including physical connectors, for example); selection of interface points; interfacing equipment; quality of service guarantees and measurement methods; charging methods and responsibilities; connectivity verification and other management tools; and regulatory issues. Many of these aspects lie outside the scope of this standard; the reader is referred to the bibliography in Annex M, which includes references to completed and ongoing work in the MEF (Metro Ethernet Forum) and the ITU.

15.1 Service transparency

The operation of Provider Bridges and the networks they compose is, by design, largely transparent to Customer Bridges and Customer Bridged Local Area Networks as illustrated by Figure 15-1.

The service provided by Provider Bridges is transparent to the use of the MAC Service by end stations attached to the Customer Bridged LANs and transparent to the operation of media access method independent functions by Customer Bridges.

The service is not transparent to the operation of media access method dependent convergence functions, specified in 6.7 of this standard, or to the operation of the media access method specific functions specified

by standards for each media access method. Media access method dependent and specific functions operate between bridges, whether Customer Bridges or Provider Bridges, attached to the same LAN. Where these functions make use of standard Group MAC Addresses, those addresses are included in the Reserved Addresses that are always filtered by Customer Bridges (Table 8-1) and by Provider Bridges (Table 8-2).

Frames transmitted and received by media access method independent functions particular to Provider Bridged Network operation are not forwarded by Customer Bridges between provider networks. Where these frames are addressed using standard Group MAC Addresses, those addresses are included in the Reserved Addresses that are always filtered by Customer Bridges (Table 8-1). In addition, such frames may be filtered from Provider Edge Ports.

15.2 Customer service interfaces

A service provider can offer a customer one or more types of service interfaces, each providing different capabilities for service selection, priority selection, and service access protection (15.7, 15.8, and 15.9). Some service interfaces are provided by the service provider operating systems that include C-VLAN components, or by customer operating systems that include S-VLAN components. In all cases, segregation of different service instances is achieved at an interface wholly under the control of the service provider by authentication and authorization of the attached customer systems, and by verification of customer provided parameters that provide service instance selection.

NOTE—The term “service access protection” describes provision of service access over multiple access LANs with redundancy and rapid failover in case of failure of an access LAN or attached equipment.

Access to a given service instance can be provided through different types of customer interface.

15.3 Port-based service interface

The customer service interfaces that can be provided by a provider network are specified by reference to a Customer Network Port provided by the S-VLAN component of a Provider Bridge. The Customer Network Port provides a single service instance, as illustrated in Figure 15-2 and Figure 15-3. The attached customer system can be a bridge, a router, or an end station.

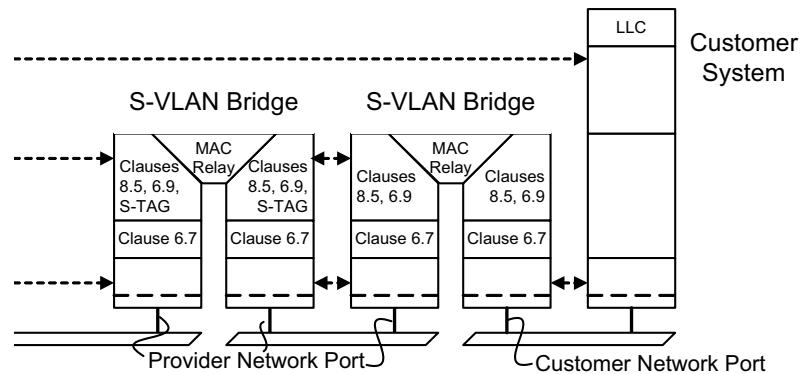


Figure 15-2—Port-based service interface to a Provider Bridged Network

This interface is Port-based; i.e., customers select and identify different service instances by associating each with a different Customer Network Port. Frames transmitted to a Customer Network Port by a C-VLAN aware customer system do not include an S-VID but can be priority-tagged with an S-TAG (6.13).

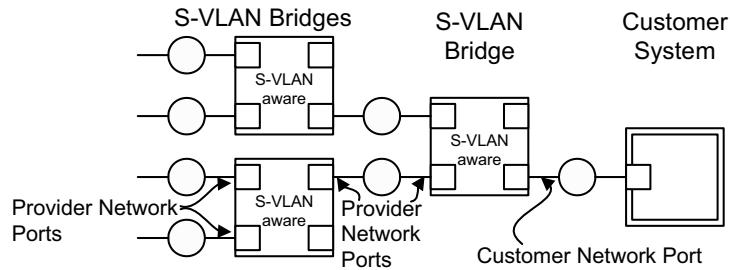


Figure 15-3—Port-based service interface to a Provider Bridged Network

NOTE—The terms “Customer Network Port,” “Customer Edge Port,” “Provider Network Port,” and “Provider Edge Port,” do not refer to the ownership of equipment, or necessarily to differently implemented Ports, but to Ports that are configured to fulfil the requirements of precise roles within a structured provider network design. These requirements are described in 15.6 through 15.9 and in Clause 16. All “Network” Ports are part of S-VLAN components, and all “Edge” Ports are part of C-VLAN components, whereas all “Customer” Ports receive data from a single customer inbound to the network and transmit data outbound from the network to a single customer. See Clause 3 for definitions.

15.4 C-tagged service interface

A C-tagged service interface can be provided by a Provider Edge Bridge comprising one or more C-VLAN components attached to Port-based service interfaces provided by a single S-VLAN component, as illustrated by Figure 15-4 and Figure 15-5.

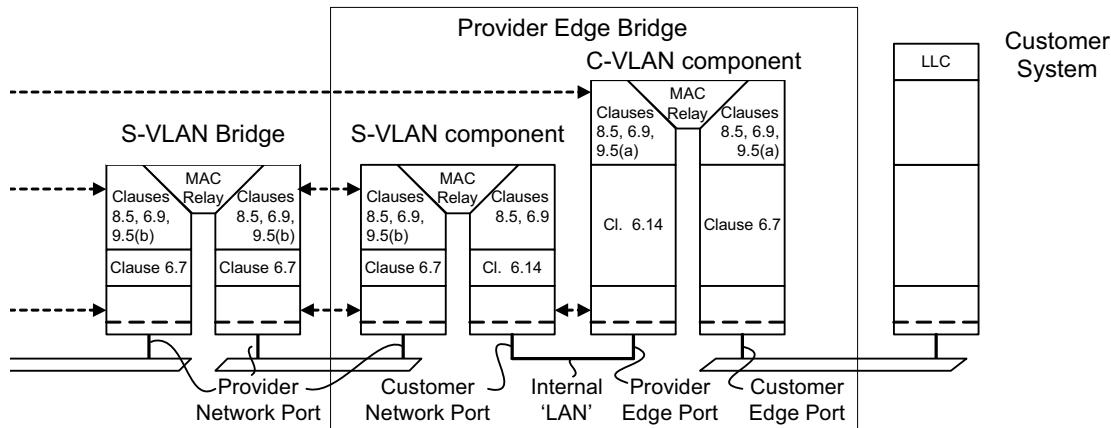


Figure 15-4—C-tagged service interface to a Provider Bridged Network

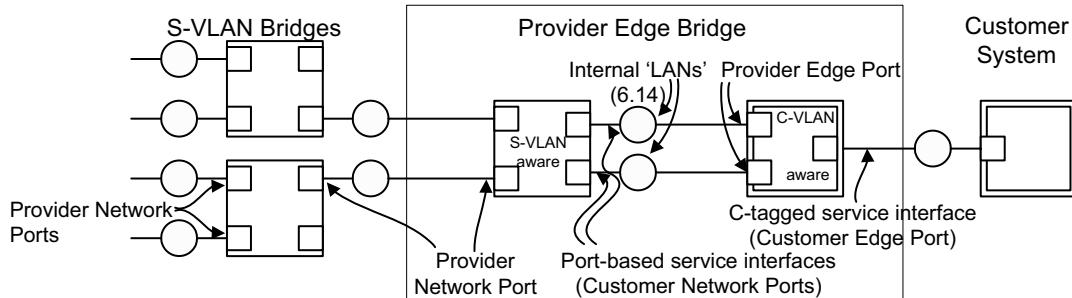


Figure 15-5—C-tagged service interface to a Provider Bridged Network

The C-tagged service interface allows service instance selection and identification by C-VID. Each frame from the customer system is assigned to a C-VLAN and presented at zero or one internal Port-based service interfaces, each supporting a single service instance that the customer desires to carry that C-VLAN.

NOTE—The restriction that each C-VLAN map to a single service instance allows the customer equipment receiving frames to correctly identify the service instance used, supports mechanisms that guard against accidental creation of data loops, and prevents configuration of the C-VLAN component to create a multi-point service from point-to-point service instances. The service provider can offer a multi-point service through appropriate configuration of the S-VLAN component.

Similarly frames from the provider network are assigned to an internal interface or “LAN” on the basis of the S-VID. As each internal interface supports a single service instance, no S-TAG is used at this interface. If multiple C-VLANs are supported by this service instance, the frames will have C-TAGs with the possible exception of frames for a single C-VLAN. The C-VLAN component applies a PVID to untagged frames received on each internal “LAN,” allowing full control over the delivery of frames for each C-VLAN through the Customer Edge Port.

Each Provider Edge Bridge can support multiple Customer Edge Ports for the same customer or for multiple customers. Each Customer Edge Port is supported by a dedicated C-VLAN component as illustrated in Figure 15-6.

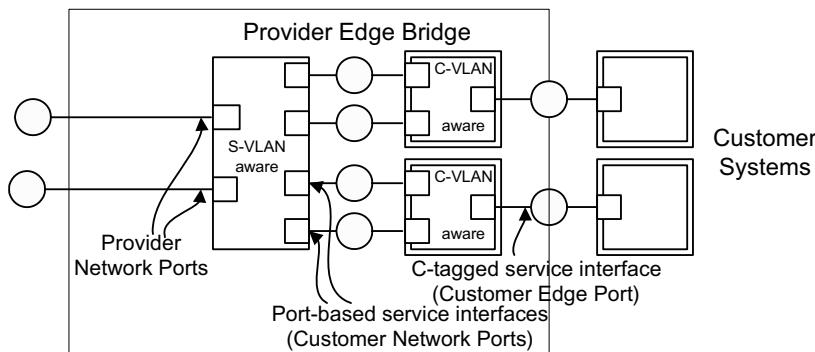


Figure 15-6—Customer Edge Ports

15.5 S-tagged service interface

An S-tagged service interface can be provided to an S-VLAN Bridge operated by a customer as illustrated by Figure 15-7 and Figure 15-8, or to a customer-operated Provider Edge Bridge that in turn provides C-tagged service interfaces within the customer’s own network as described in 15.4.

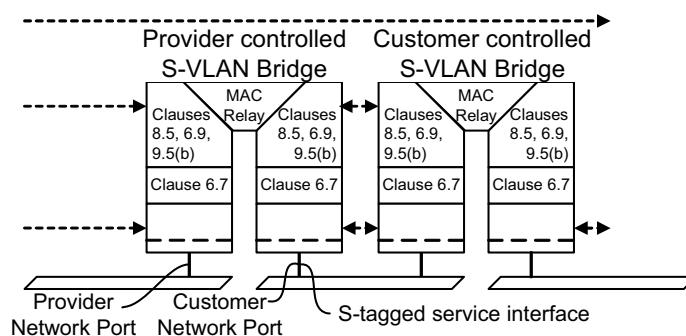


Figure 15-7—S-tagged service interface to a Provider Bridged Network

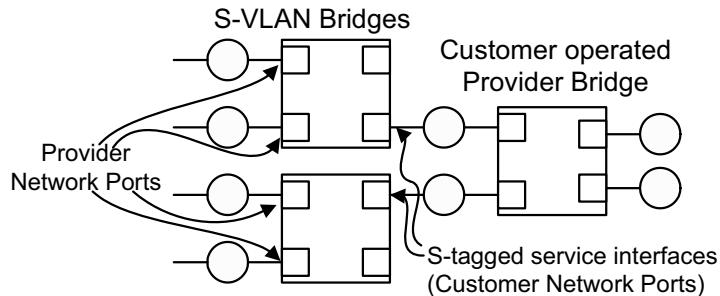


Figure 15-8—S-tagged interface to a Provider Bridged Network

15.6 Service instance segregation

Segregation of data frames associated with different MAC Service instances is achieved by supporting each service instance with a separate Service VLAN (S-VLAN) and ensuring that:

- a) Provider Bridges are configured such that no customer data frames are transmitted through a Provider Network Port untagged, i.e., without a Service VLAN Tag (S-TAG).
- b) No frames are accepted, i.e., received and relayed, from any customer system without first being subject to service instance selection.
- c) No frames are delivered to any customer system without explicit service instance identification.
- d) Prior to transmission through a Provider Network Port, customer data frames are received through a Customer Network Port within the provider network that is exclusively accessed by a single customer. The S-VIDs of all frames received through that Customer Network Port correspond to service instances that the customer is permitted to access.
- e) Provider Bridges and the S-VLAN component of each Provider Edge Bridge within the provider network can only be directly controlled by the service provider. Customer equipment, including customer-owned Provider Bridges, are not within the provider network and are controlled by the customer.
- f) Only frames that have been transmitted through a Provider Network Port can be received through other Provider Network Ports within the provider network.

15.7 Service instance selection and identification

Service instance selection is provided for Port-based service interfaces by configuring a Customer Network Port with a PVID value corresponding to the S-VID used to identify the service instance and an Acceptable Frame Types value of *Admit Only Untagged and Priority-tagged frames*.

Service instance selection is provided for C-tagged service interfaces by a C-VLAN component internal to a Provider Edge Bridge. The C-VLAN component uses the C-VID to direct frames to an internal Provider Edge Port supporting a specific service instance. Frames for at most one C-VLAN can be conveyed untagged over a single service instance. Management control of associating C-VIDs with Provider Edge Ports is accomplished using the C-VID Registration Table (12.13.3), which provides equivalent functionality to configuring the PVID of the internal Customer Network Port with the S-VID of the service instance, and adding the Provider Edge Port to the Member Set, and possibly Untagged Set, of the C-VLAN. No management control is provided for Protocol-based VID assignment on internal Customer Network Ports.

NOTE 1—A Provider Edge Bridge can configure the C-VLAN component associated with a Customer Edge Port to select the same service instance for all frames. This creates a service interface similar, but not identical, to a Port-based service interface. The C-VLAN component allows modification of the C-TAG (insertion of a C-TAG with the PVID

value in untagged frames, assigning the PVID value to priority-tagged frames, or stripping the C-TAG from frames forwarded through the Provider Edge Port) but never forwards a frame with a null C-VID. A Port-based service interface does not modify the C-TAG of received frames in any way. A Provider Edge Bridge may offer a Port-based service interface by configuring the Port to be a Customer Network Port rather than a Customer Edge Port; in which case, there is no associated C-VLAN component.

Service instance selection is provided by the attached customer system for S-tagged service interfaces. The Customer Network Port is configured with Enable Ingress Filtering (8.6.2), and the Port is only included in the Member Set for S-VLANs corresponding to service instances that the customer is permitted to use.

For all service interfaces described, the Customer Network Port determines the S-VID for each customer data frame as specified in 6.9.1 for an EISS instance using a Service VLAN Tag type.

The VID Translation Table for the Port (6.9.1) allows a service provider to assign S-VIDs independently from those used by a customer (or other service provider) to identify service instances on an S-tagged service interface (15.5). The table also allows customers to identify the same service instance by different VIDs at different interfaces.

NOTE 2—The means used by a service provider and a customer to determine the VIDs used by the customer to select and identify a given service instance are outside the scope of this standard.

The service instance for each frame received by the attached customer system is identified in the same way as frames transmitted using the same interface, but not necessarily in the same way that the service instance is selected or identified at other interfaces. A single service instance can support Port-based, C-tagged, and S-tagged service interfaces.

15.8 Service priority selection

For all service interface types, the service priority is selected using the received priority for each frame, possibly regenerated using the Priority Regeneration Table (6.9.4). The mechanism for determining the received priority varies with the type of service interface.

Service priority selection is provided for Port-based service interfaces using the received priority signaled from the media access method of the port. If the media access method used to attach to the interface does not directly support priority, this will result in the selection of a single value for all frames. A customer system may also signal priority to a Port-based service interface on a per-frame basis by priority-tagging each frame with an S-TAG with a null VID. Subclause 6.13 specifies a function to support priority-tagging with an S-TAG on Customer Bridges.

Service priority selection is provided by C-tagged service interfaces using the priority conveyed in the C-TAG of each frame. A C-tagged service interface can provide a single service instance for all C-VIDs received and in this way function much as a Port-based service interface with the addition of the capability of the customer to independently signal priority with each frame.

Service priority selection is provided by S-tagged interfaces using the received priority decoded from the PCP field in the S-TAG.

15.9 Service access protection

A customer system or systems at a single location can attach to two or more service interfaces using separate LANs for attachment, thus providing fault tolerance through redundancy of the interface components.

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.

15.11 Data-driven and data-dependent connectivity fault management (DDCFM)

Data-driven and data-dependent connectivity fault management (DDCFM), described in Clause 29, provides tools for network operators to detect and isolate data-driven and data-dependent faults in Virtual Bridged Local Area Networks.

16. Principles of Provider Bridged Network operation

This clause establishes the principles and a model of Provider Bridged Network operation. It provides the context necessary to understand how the

- a) Operation of individual Provider Bridges (Clause 8),
- b) Configuration and management of individual Provider Bridges (Clause 12), and
- c) Management of Spanning Tree and VLAN Topologies within a provider network (Clause 7, Clause 11, Clause 13)

support, preserve, and maintain the quality of each instance of the MAC Service offered to the customers of the provider network (Clause 6, Clause 15), including

- d) Independence of each service instance supported by a service provider from other service instances (Clause 15);
- e) Identification of service instances within the provider network (Clause 15, 8.8);
- f) Maintenance of service availability in the event of the failure, restoration, removal, or insertion of LAN components connecting a customer network to a provider network (Clause 6, Clause 11, Clause 13, 16.2).

A Provider Bridged Network is a Virtual Bridged Local Area Network that comprises Provider Bridges (S-VLAN Bridges and Provider Edge Bridges) and attached LANs, under the administrative control of a single service provider. The principal elements of provider network operation are those specified in Clause 7 for Virtual Bridged Local Area Networks in general, as amended by this clause.

NOTE 1—Unless explicitly stated, the term “provider network” in this standard refers to a Provider Bridged Network. The term “Provider Bridged Network” is used exclusively to refer to networks configured and managed as specified by this clause and comprising only (a) Provider Bridges and Provider Edge Bridges and (b) communications media and equipment providing the Internal Sublayer Service (6.6). Although the requirements of Clause 15 are generally applicable to similar services, a generalized framework for all network designs that could support these requirements, while useful in the context of other equipment and services, is outside the scope of this standard. This clause describes specific best practice for Provider Bridged Networks to ensure that the requirements for bridge functionality are clear. Conformance of a Provider Bridge implementation to this standard does not require that the implementation be used as specified in this clause, merely that it is capable of being so used.

NOTE 2—Within a provider network, an instance or instances of the MAC Service are reserved for the service provider’s own use to configure and manage the network. All frames associated with such service instances, and that are not confined to an individual LAN, are subject to service instance selection, segregation, and identification as specified in 16.1.

16.1 Provider Bridged Network overview

The principal elements of Provider Bridge Network operation comprise

- a) Service instance segregation within the provider network for customer frames (15.6).
- b) Service instance selection on ingress, and service instance identification on egress, for each customer frame (15.7).
- c) Resource allocation and configuration to provide service instance connectivity (16.3).

and may also include

- d) Management of customer end station address learning (16.4).
- e) Prevention of connectivity loops formed through attached networks (16.5).

16.2 Provider Bridged Network

An example Provider Bridged Network is illustrated by Figure 16-1.

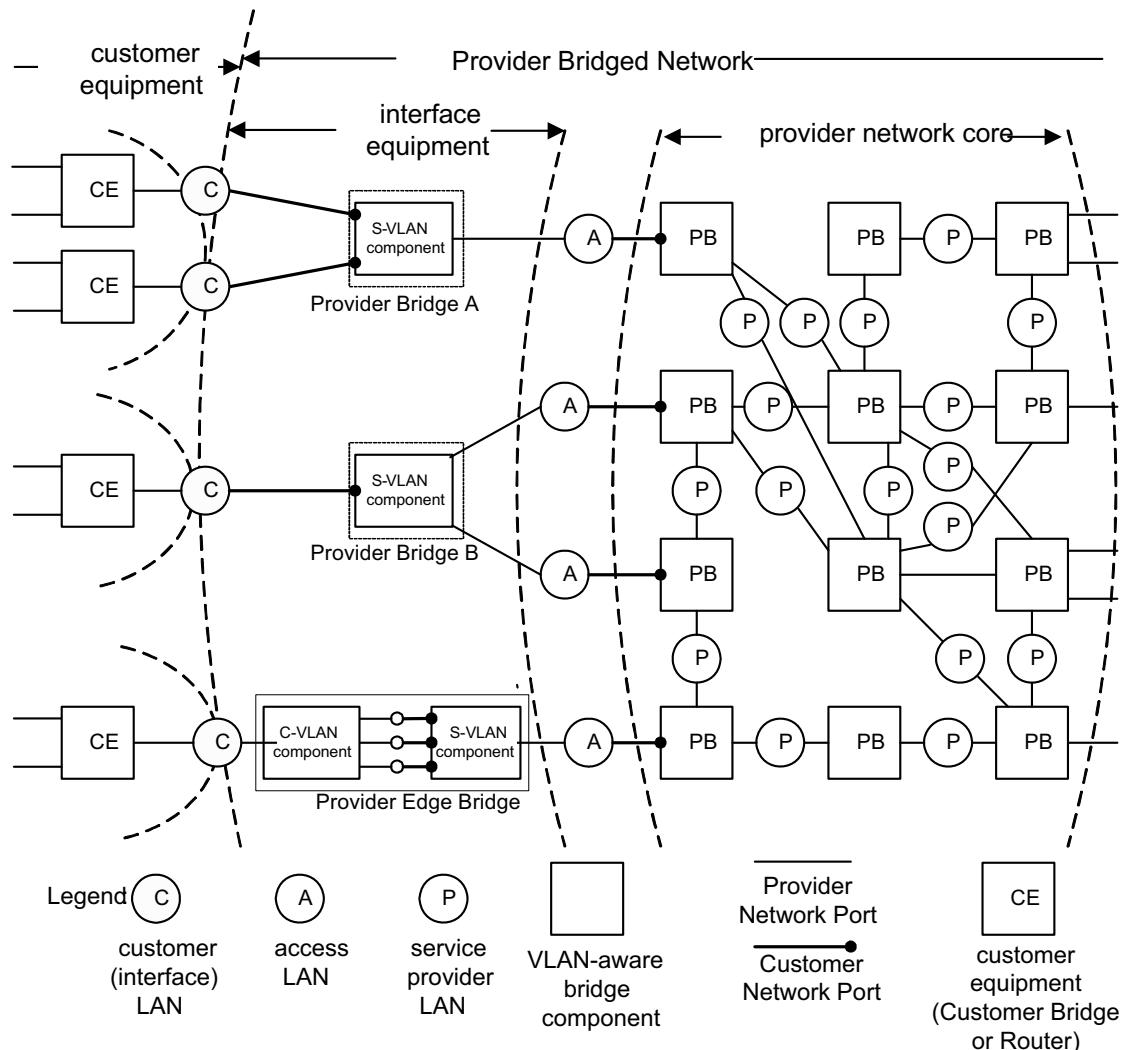


Figure 16-1—Provider Bridged Network with interface examples

Customer equipment attaches, via one or more customer interface LANs, to equipment that provides the service interfaces specified in Clause 15. That interface equipment can be located in the core of the provider network, such that both the equipment and the attached LANs are secure against direct addition of frames either by the customer or by others. More commonly the interface equipment connects to the network core using one or more access LANs, which can be subject to external interference. Figure 16-1 provides examples. Within the network core, Provider Bridges and LANs are secured so that only the service provider can manage the reception, transmission, and relay of frames between Provider Bridges.

The arbitrary physical network topology of the network core and the connectivity that it provides to support segregated instances (15.6) of the MAC Service is designed and managed (16.3) by the service provider to meet bandwidth and service availability requirements at the Provider Network Ports. Application of the Service VLAN ingress and egress rules at these Ports in support of service instance selection and identification (15.7) ensures that frames cannot be transmitted or received on any service instance by any customer's equipment without prior agreement with the service provider.

Although the application of the ingress and egress rules, together with the use of the MSTP restrictedRole and restrictedTcn (13.25.50, 13.26.21) parameters and MVRP registration controls (16.3), permit service providers to allow direct attachment of customer-operated equipment to access LANs, there are commonly other reasons, such as OAM&P support of access LANs, why interface equipment is mandated. The interface equipment, as illustrated by, but not limited to, the examples in Figure 16-1 can be used to partition and enhance network access functionality to provide

- a) Service instance multiplexing on a single access LAN;
- b) Provision of resilient, and optionally physically route diverse, access;
- c) Reduced management of customer use of multiple service instances;
- d) Selective multiplexing of customer VLANs onto service instances;
- e) Reliable identification of the customer point of attachment;

without requiring customer systems to understand the internal details of the provider network.

Provider Bridge A in Figure 16-1 uses physically separate customer interface LANs to provide separate Port-based service interfaces to the customer equipment and uses Service VLAN tags to multiplex the corresponding service instances over a single access LAN. The interface equipment can be managed by the service provider to use S-VIDs that do not require use of a VID Translation Table at the edge of the core network. Alternatively, the latter can translate S-VIDs to remove the need for such management, with all interface equipment using the same S-VIDs in the same way.

Provider Bridge B in Figure 16-1 uses a single customer interface LAN to provide a Port-based service interface to the customer equipment and uses two access LANs to provide resilient connectivity to distinct Provider Bridges in the core network. Receipt of Provider Bridge BPDUs by Bridge B protects against the failure of an access LAN, of one of the core bridges, or of some internal physical connectivity within the provider network. Use of the MSTP restrictedRole and restrictedTcn parameters by Provider Network Ports ensures that receipt of frames from the access LANs cannot disrupt the active topology or address learning within the core network. Where multiple service instances are provided at a single customer point of attachment, both access links can be used.

Provider Edge Bridge C in Figure 16-1 provides a C-tagged service interface to the customer equipment and uses Customer VLAN tags on the customer interface LAN to select between multiple Service VLAN tagged service instances on the access LAN. Individual Customer VLANs can be conveyed on any service instance, and this distribution of Customer VLANs can be changed in response to changes in the connectivity offered by the service instances through customer systems at other points of customer attachment. The traffic for a particular Customer VLAN can be the majority of that carried on a specific service instance; specifying that frames for that Customer VLAN are untagged on the internal LAN that corresponds to that service instance within the Provider Edge Bridge allows those frames to be carried through the provider network without a C-VID following the S-VID if the overhead of conveying the additional octets is a concern.

NOTE—In the scenario where a customer network has multiple service interfaces to a provider network, and the customer network VLAN topology is such that frames sent from a single MAC source on different C-VLANs will enter the provider network on different service interfaces and be mapped to the same service instance (same S-VLAN), then the customer MAC address will appear to be duplicated and potentially cause oscillation in the learning process within the Provider Network. This scenario can be avoided by restricting a MAC address to enter the provider network at the same service interface for all C-VLANs, or by mapping the different C-VLANs to different service instances (different S-VLANs), or by restricting the service instance to be point-to-point so the provider bridges do not need to learn the customer MAC addresses.

16.3 Service instance connectivity

The VLAN Topology of each S-VLAN is established by the mechanisms introduced in 7.1 and Figure 7-1. The service provider can use and configure MSTP to provide a number of independent spanning tree active

topologies and can assign each S-VLAN independently to one of these to best use the resources in the network. MVRP running in the context of each spanning tree active topology configures the extent of each S-VLAN to the subset of that active topology necessary to support connectivity between the customer points of attachment to the MAC Service instance provided, and it can reconfigure that connectivity as required if the spanning tree active topology changes.

NOTE 1—Autoconfiguration of the extent of each S-VLAN is accomplished by the service provider configuring the MVRP Administrative Control “Registration Fixed” for the S-VLAN on each Customer Network Port where the corresponding MAC Service Instance can be selected. The Enable Ingress Filtering parameter is not typically used within an individual provider network, as it limits the ability of the network to carry service instances following changes in the active topology. However, it can be used to limit the reachability of service instances used by the service provider for network management and to restrict service instances carried from one provider network domain to another.

The operation of MSTP within a provider network is independent of the operation of any spanning tree protocol within attached customer networks. This independence is achieved by using the Provider Bridge Group Address (Table 8-1) as the destination address of all MSTP BPDUs transmitted on all Provider Bridge Ports and by setting the restrictedRole and restrictedTcn parameters (13.25.50, 13.26.21) for Customer Network Ports. Frames received by Customer Network Ports and addressed to the Bridge Group Address are subject to service instance selection and relay in the same way as customer data frames.

NOTE 2—Customer BPDUs addressed to the Bridge Group Address are conveyed transparently, allowing the customer to use an instance of MSTP or RSTP that is completely independent of the provider network to establish and maintain full and loop-free connectivity of the customer connected networks and services. A customer can also use MVRP to limit the transmission of frames assigned to C-VLANs to the service instances required for C-VLAN connectivity.

The operation of MVRP within a provider network is independent of the operation of any configuration protocol within attached customer networks. The Provider Bridge MVRP Address (Table 8-1) is used as the destination address of all MVRPDUs transmitted in support of the MVRP Application. Frames received by Customer Network Ports and addressed to an MRP Application Address (Table 10-1) not in use by the S-VLAN component are subject to service instance selection and relay in the same way as customer data frames. The MVRP Administrative Control for each S-VLAN is either “Registration Fixed” or “Registration Forbidden” on all Customer Network Ports, so no information is received from any Provider Bridge MVRPDU that has been erroneously transmitted by a customer system.

16.4 Service provider learning of customer end station addresses

Customer data frames for any given MAC Service instance are restricted to that part of the provider network that supports the VLAN topology of the associated S-VLAN as described in 16.3 and are further restricted by learning the source addresses of frames as described in Clause 7 and Clause 8.

In a Provider Bridged Network that commonly provides interfaces to each customer at a small fraction of the total number of customer interfaces provided, the requirement for learning customer end station addresses can be much reduced by applying enhanced filtering utility criteria (8.7). In particular, learning can be restricted to the ingress and egress Provider Bridge Ports of each S-VLAN that connects only two customer points of attachment, or to the customer systems attached to those Ports.

16.5 Detection of connectivity loops through attached networks

The transmission and reception of MSTP BPDUs through Customer Network Ports will detect accidental direct connection of those ports, or their interconnection by a network that is transparent to frames with the Provider Bridge Group Address as the destination MAC address. However, a service provider cannot rely on any customer network relaying such frames and should develop a policy and mechanisms to deal with potential data loops that can arise if the attached customer systems do not correctly operate their own instance or instances of a spanning tree protocol.

NOTE 1—Use of the restrictedRole parameter at ingress ports ensures that receipt of BPDUs addressed to the Provider Bridge Group Address cannot disrupt internal connectivity within the provider network.

NOTE 2—Specification of service provider policies, mechanisms, and heuristics used to detect or minimize the impact of data loops created by customer systems is not addressed by this standard. They can include, but are not limited to, bandwidth limitation, charging policies, detection of the repetitive movement of the apparent location of customer stations, and customer agreement to allow the use of service provider loop detection protocols by not filtering the associated frames.

NOTE 3—A data loop is not the only possible cause of excess bandwidth consumption by a given customer of a provider network, and the service provider is usually required to meet service guarantees to other customers irrespective of the cause of the excess bandwidth demand. Data loops are not a unique threat to satisfactory overall network performance. Their distinct characteristic is consumption of discretionary bandwidth without benefitting any customer. The customer that creates the loop can suffer particularly serious network degradation or excess cost as the service provider limits the total bandwidth consumed by that customer. It is, therefore, in the interests of each customer and the service provider to raise service satisfaction by preventing and detecting loops.

Each C-VLAN component within a Provider Edge Bridge implements RSTP, with the enhancements to support loop-free connectivity between Customer Edge Ports, as specified in 13.39, thus interoperating with customer equipment and spanning tree configurations to support reliable and deterministic prevention of loops, and supporting provision of redundant connectivity.

16.6 Network management

Management of a Provider Bridge is directly under the control of the service provider. Provider network customers shall not have access to managed objects related to elements of Provider Bridges within the provider network.

17. Management Information Base (MIB)

The clause contains a complete SMIv2 Management Information Base (MIB) set for all features of this standard. Previous versions of this standard (e.g., IEEE Std 802.1Q-2005, IEEE Std 802.1Q-1998) relied on MIB modules defined by the IETF Bridge Working Group. IETF RFC 4663 (2006) [B31] describes more details of the IETF transition of responsibility for bridging-related MIB modules from the IETF Bridge MIB Working Group to the IEEE 802.1 Working Group.

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).

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 IETF STD 58, IETF RFC 2578 (1999), IETF RFC 2579 (1999), and IETF RFC 2580 (1999).

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 IETF 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 MIB modules in 17.7.

17.2 Structure of the MIB

The IEEE 802.1Q MIB is divided into a number of modules corresponding to the project that created the functionality. The managed objects in most of these modules have been reindexed to allow their use within the PBB MIB module (and thus avoid a duplication of objects).

A summary of the modules contained in this clause is presented in Table 17-1.

Table 17-1—Structure of the MIB modules

Module	Subclause	Defining IEEE standard	Reference	Notes
IEEE8021-TC MIB	17.7.1	802.1ap	—	Textual conventions for all modules
IEEE8021-BRIDGE MIB	17.7.2	802.1D, 802.1p, and 802.1t	802.1D	Adapted from IETF RFC 4188 and IETF RFC 4363
IEEE8021-SPANNING-TREE MIB	17.7.3	802.1w	802.1D 17	Adapted from IETF RFC 4318
IEEE8021-Q-BRIDGE MIB	17.7.4	802.1Q, 802.1u, and 802.1v	8	Adapted from IETF RFC 4363
IEEE8021-PB MIB	17.7.5	802.1ad	16	Initial version in IEEE Std 802.1ap
IEEE8021-MSTP MIB	17.7.6	802.1s	13	Initial version in IEEE Std 802.1ap

Table 17-1—Structure of the MIB modules (continued)

Module	Subclause	Defining IEEE standard	Reference	Notes
IEEE8021-CFM MIB	17.7.7.1	802.1ag	20	Initial version in IEEE Std 802.1ag
IEEE8021-CFM MIBV2	17.7.7.2	802.1ag	20	Initial version in IEEE Std 802.1ap
IEEE8021-PBB MIB	17.7.8	802.1ah	26	Initial version in IEEE Std 802.1ap
IEEE8021-DDCFM MIB	17.7.9	802.1Qaw	29	Initial version of IEEE Std 802.1Qaw
IEEE8021-PBBTE MIB	17.7.10	802.1Qay	—	Initial version in IEEE Std 802.1Qay
IEEE8021-TPMR MIB	17.7.11	802.1aj	5.12, 5.13,	Initial version in IEEE Std 802.1aj
IEEE8021-FQTSS MIB	17.7.12	802.1Qav	34	Initial version in IEEE Std 802.1Qav
IEEE8021-CN-MIB	17.7.13	802.1Qau	30	Initial version in IEEE Std 802.1Qau
IEEE8021-SRP MIB	17.7.14	802.1Qat	35	Initial version in IEEE Std 802.1Qat

17.2.1 Structure of the IEEE8021-TC MIB

The purpose of the IEEE8021-TC MIB is to define textual conventions used throughout the IEEE 802.1Q MIB modules. Textual conventions (TCs) originally appeared in each of the MIB modules for IEEE Std 802.1Q produced by the IETF. However, with the transition of all modules for IEEE Std 802.1Q to this clause, it made sense to have a single module with all the textual conventions contained within it. Note that many of the original IETF TCs are still used in the various MIB modules for VLAN-Bridge management and they have not been imported here.

NOTE—The IEEE8021-FQTSS MIB module in 17.7.12 defines additional TEXTUAL-CONVENTIONS that are utilized by the managed objects that support use of the Credit-based shaper algorithm (8.6.8.2, 34).

The textual conventions contained in this IEEE8021-TC module are summarized in Table 17-2.

Table 17-2—IEEE8021-TC MIB Structure

IEEE MIB object	Reference
IEEE8021PbbComponentIdentifier	12.3.1)
IEEE8021PbbComponentIdentifierOrZero	12.3.1)
IEEE8021PbbServiceIdentifier	12.16.3, 12.16.5
IEEE8021PbbServiceIdentifierOrUnassigned	12.16.3, 12.16.5
IEEE8021PbbIngressEgress	12.16.3, 12.16.5
IEEE8021PriorityCodePoint	12.6.2

Table 17-2—IEEE8021-TC MIB Structure (continued)

IEEE MIB object	Reference
IEEE8021BridgePortNumber	12.3 i), 17.3.2.2
IEEE8021BridgePortNumberOrZero	12.3 i), 17.3.2.2
IEEE8021BridgePortType	12.16.1.1.3 h4), 12.16.2.1, 12.13.1.1, 12.13.1.2
IEEE8021VlanIndex	9.6
IEEE8021VlanIndexOrWildcard	9.6
IEEE8021MstIdentifier	13.7
IEEE8021ServiceSelectorType	—
IEEE8021ServiceSelectorValueOrNone	—
IEEE8021ServiceSelectorValue	—
IEEE8021PortAcceptableFrameTypes	12.10.1.3, 12.13.3.3, 12.13.3.4
IEEE8021PriorityValue	12.13.3.3
IEEE8021PbbTeProtectionGroupId	12.19.2
IEEE8021PbbTeEsp	3.63
IEEE8021PbbTeTsidId	3.190
IEEE8021PbbTeProtectionGroupConfigAdmin	26.10.3.3.4, 26.10.3.3.5, 26.10.3.3.6, 26.10.3.3.7, 12.18.2.1.3 d)
IEEE8021PbbTeProtectionGroupActiveRequests	12.18.2.3.2

The textual conventions used by IEEE 802.1Q MIB modules not contained in this TC module, but contained in IETF RFC 4318 and IETF RFC 4363 are as follows:

- BridgeId
- Timeout
- PortList
- VlanIdOrAny
- VlanIdOrNone
- VlanIdOrAnyOrNone

17.2.2 Structure of the IEEE8021-BRIDGE MIB

The work in this standard is based upon the BRIDGE MIB version in IETF RFC 4188 and the P-BRIDGE MIB originally in IETF RFC 4363. While many tables and objects are similar, this IEEE8021-BRIDGE MIB has been reindexed and rerooted with its inclusion in this clause.

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. The overall structure and assignment of objects to their subtrees is shown below. Where appropriate, the corresponding IETF RFC 4188 object name, IEEE 802.1D management object name, and reference is also

included. If the IETF RFC mapping is missing, it means there is no equivalent in the old MIBs. If the Managed Object is missing, it means that no managed object is specified in the referenced clause.

Table 17-3 indicates the structure of the IEEE8021-BRIDGE MIB module as well as showing its relationship to the IETF MIB that is being obsoleted.

Table 17-3—IEEE8021-BRIDGE MIB structure and relationship to IETF RFC 4188 and this standard

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021BridgeBaseTable	dot1dBase	12.4.1
ieee8021BridgeBaseComponentId [*]	—	—
ieee8021BridgeBaseBridgeAddress	dot1dBaseBridgeAddress	12.4.1.1.3 a)
ieee8021BridgeBaseNumPorts	dot1dBaseNumPorts	12.4.1.1.3 c)
ieee8021BridgeBaseComponentType	—	12.3 m)
ieee8021BridgeBaseDeviceCapabilities	dot1dDeviceCapabilities	12.10.1.1.3 b)
ieee8021BridgeBaseTrafficClassesEnabled	dot1dTrafficClassesEnabled	—
ieee8021BridgeBaseMmrpEnabledStatus	dot1dGmrpStatus	—
ieee8021BridgeBaseRowStatus		
ieee8021BridgeBasePortTable	dot1dBasePortTable	12.4.2
ieee8021BridgeBasePortComponentId [*]	—	—
ieee8021BridgeBasePort [*]	dot1dBasePort	12.4.2.1.2 a)
ieee8021BridgeBasePortIfIndex	dot1dBasePortIfIndex	—
ieee8021BridgeBasePortDelayExceededDiscards	dot1dBasePortDelayExceededDiscards	12.6.1.1.3 f)
ieee8021BridgeBasePortMtuExceededDiscards	dot1dBasePortMtuExceededDiscards	12.6.1.1.3 g)
ieee8021BridgePortCapabilities	dot1dPortCapabilities	12.10.1.1.3 c)
ieee8021BridgeBasePortTypeCapabilities		—
ieee8021BridgeBasePortType		—
ieee8021BridgeBasePortExternal		—
ieee8021BridgeBasePortAdminPointToPoint	dot1dStpPortAdminPointToPoint	12.8.2.1.3 o)
ieee8021BridgeBasePortOperPointToPoint	dot1dStpPortOperPointToPoint	12.8.2.1.3 p)
ieee8021BridgeBasePortName	ifDescr	12.4.2.1.3 a)
ieee8021BridgeTpPortTable	dot1dTpPortTable	12.4.2, C.4
ieee8021BridgeTpPortComponentId [*]	—	—

Table 17-3—IEEE8021-BRIDGE MIB structure and relationship to IETF RFC 4188 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021BridgeTpPort*	dot1dTpPort	—
ieee8021BridgeTpPortMaxInfo	dot1dTpPortMaxInfo	—
ieee8021BridgeTpPortInFrames	dot1dTpPortInFrames	12.6.1.1.3 a)
ieee8021BridgeTpPortOutFrames	dot1dTpPortOutFrames	12.6.1.1.3 d)
ieee8021BridgeTpPortInDiscards	dot1dTpPortInDiscards	12.6.1.1.3 c)
ieee8021BridgePortPriorityTable	dot1dPortPriorityTable	12.6.2
(AUGMENTS ieee8021BridgeBasePortEntry)	(AUGMENTS dot1dBasePortEntry)	—
ieee8021BridgePortDefaultUserPriority	dot1dPortDefaultUserPriority	—
ieee8021BridgePortNumTrafficClasses	dot1dPortNumTrafficClasses	—
ieee8021BridgePortPriorityCodePointSelection		12.6.2.6, 12.6.2.7
ieee8021BridgePortUseDEI		12.6.2.12, 12.6.2.13
ieee8021BridgePortRequireDropEncoding		12.6.2.14, 12.6.2.15
ieee8021BridgePortServiceAccessPrioritySelection		12.6.2.16, 12.6.2.17
ieee8021BridgeUserPriorityRegenTable	dot1dUserPriorityRegenTable	6.6
ieee8021BridgeUserPriorityRegenComponentId*		—
ieee8021BridgeBasePort*	dot1dBasePort	—
ieee8021BridgeUserPriority*	dot1dUserPriority	—
ieee8021BridgeRegenUserPriority	dot1dRegenUserPriority	—
ieee8021BridgeTrafficClassTable	dot1dTrafficClassTable	Table 8-4
ieee8021BridgeTrafficClassComponentId*		—
ieee8021BridgeBasePort*	dot1dBasePort	—
ieee8021BridgeTrafficClassPriority*	dot1dTrafficClassPriority	—
ieee8021BridgeTrafficClass	dot1dTrafficClass	—
ieee8021BridgePortOutboundAccessPriorityTable	dot1dPortOutboundAccessPriorityTable	Table 8-4
ieee8021BridgePortOutboundAccessPriorityComponentId*		—

Table 17-3—IEEE8021-BRIDGE MIB structure and relationship to IETF RFC 4188 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021BridgeBasePort [*]	dot1dBasePort	—
ieee8021BridgePortOutboundAccessPriority [*]	dot1dPortOutboundAccessPriority	—
ieee8021BridgePortDecodingTable		12.6.2
ieee8021BridgePortDecodingComponentId [*]		—
ieee8021BridgePortDecodingPortNum [*]		—
ieee8021BridgePortDecodingPriorityCodePointRow [*]		—
ieee8021BridgePortDecodingPriorityCodePoint [*]		—
ieee8021BridgePortDecodingPriority		12.6.2.7, 12.6.2.8
ieee8021BridgePortDecodingDropEligible		12.6.2.12, 12.6.2.13
ieee8021BridgePortEncodingTable		12.6.2
ieee8021BridgePortEncodingComponentId [*]		—
ieee8021BridgePortEncodingPortNum [*]		—
ieee8021BridgePortEncodingPriorityCodePointRow [*]		—
ieee8021BridgePortEncodingPriorityCodePoint [*]		—
ieee8021BridgePortEncodingDropEligible [*]		—
ieee8021BridgePortEncodingPriority		12.6.2.9, 12.6.2.10
ieee8021BridgeServiceAccessPriorityTable		—
ieee8021BridgeServiceAccessPriorityComponentId [*]		—
ieee8021BridgeServiceAccessPriorityPortNum [*]		—
ieee8021BridgeServiceAccessPriorityReceived [*]		—
ieee8021BridgeServiceAccessPriorityValue		12.6.2.16, 12.6.2.17
ieee8021BridgePortMrpTable	dot1dPortGarpTable	12.9
(AUGMENTS ieee8021BridgeBasePortEntry)	(AUGMENTS dot1dBasePortEntry)	12.9.1.1, 12.9.1.2
ieee8021BridgePortMrpJoinTime	dot1dPortGarpJoinTime	—
ieee8021BridgePortMrpLeaveTime	dot1dPortGarpLeaveTime	—

Table 17-3—IEEE8021-BRIDGE MIB structure and relationship to IETF RFC 4188 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021BridgePortMrpLeaveAllTime	dot1dPortGarpLeaveAllTime	—
ieee8021BridgePortmMrpTable	dot1dPortGmrpTable	12.11
(AUGMENTS ieee8021BridgeBasePortEntry)	(AUGMENTS dot1dBasePortEntry)	—
ieee8021BridgePortmMrpEnabledStatus	dot1dPortGmrpStatus	—
ieee8021BridgePortMmrpFailedRegistrations	dot1dPortGmrpFailedRegistrations	—
ieee8021BridgePortMmrpLastPduOrigin	dot1dPortGmrpLastPduOrigin	—
ieee8021BridgePortRestrictedGroupRegistration	dot1dPortRestrictedGroupRegistration	12.11.1.3
ieee8021BridgeILanIfTable		17.3.2.2
ieee8021BridgeILanIfRowStatus		—
ieee8021BridgeDot1dPortTable		17.5.3
ieee8021BridgeBasePortComponentId [*]		—
ieee8021BridgeBasePort [*]		—
ieee8021BridgeDot1dPortRowStatus		—

*This object is an INDEX of the table in which it resides.

The IEEE8021-BRIDGE MIB contains the following object subtrees:

- a) The ieee8021BridgeBase Subtree
This subtree contains the objects that are applicable to all types of Bridges.
- b) The ieee8021BridgeTp Subtree
This subtree contains objects that describe the entity's state with respect to transparent bridging. If transparent bridging is not supported, this subtree will not be implemented. This subtree is applicable to transparent-only Bridges.
- c) The ieee8021BridgePriority Subtree
This subtree contains the objects for configuring and reporting status of priority-based queuing mechanisms in a bridge. This includes per Bridge Port user_priority treatment, mapping of user_priority in frames into internal traffic classes, and outbound user_priority and access_priority.
- d) The ieee8021BridgeMrp Subtree
This subtree contains the objects for configuring and reporting on operation of the Multiple Registration Protocol (MRP).
- e) The ieee8021BridgeMmrp Subtree
This subtree contains the objects for configuring and reporting on operation of the MRP MAC Registration Protocol (MMRP).
- f) The ieee8021BridgeInternal LAN Subtree
This subtree contains the objects for configuring and reporting on operation of the Internal LAN interface. An I-LAN Interface is used to create internal connections between Bridge Ports in an IEEE 802.1 device.

The IEEE 802.1D management objects in Table 17-4 have not been included in the BRIDGE-MIB module for the indicated reasons. Note that there is a conformance requirement that the base modules SNMPv2-MIB [RFC3418] and IF-MIB [RFC2863] will be implemented as a default on all Bridges.

Table 17-4—IEEE 802.1D objects not in the IEEE8021-BRIDGE MIB

IEEE 802.1D object	Disposition
Bridge.BridgeName	Same as sysDescr (SNMPv2-MIB)
Bridge.BridgeUpTime	Same as sysUpTime (SNMPv2-MIB)
Bridge.PortAddresses	Same as ifPhysAddress (IF-MIB)

17.2.3 Structure of the IEEE8021-SPANNING-TREE MIB

A RSTP MIB originally appeared in IETF RFC 4318. While many tables and objects are similar, the IEEE8021-SPANNING-TREE MIB has been merged with spanning tree objects from the original BRIDGE MIB in IETF RFC 4188 and also reindexed and rerooted with its inclusion in this subclause.

This subclause defines an SMIV2 MIB module for managing the Rapid Spanning Tree Protocol (RSTP) capability originally defined by the IEEE Std 802.1t™ and IEEE Std 802.1w™ amendments to IEEE Std 802.1D, 1998 Edition, for bridging between LANs. The objects in this MIB are defined to apply both to transparent bridging and to Bridges connected by subnetworks other than LANs.

The IEEE8021-RSTP MIB in this clause, however, is based on IEEE Std 802.1D-2004, though the references are to this standard.

Table 17-5 indicates the structure of the IEEE8021-RSTP MIB module as well as showing its relationship to the IETF MIB that is being obsoleted.

Table 17-5—IEEE8021-SPANNING-TREE MIB structure and relationship to IETF RFC 4318 and this standard

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021SpanningTreeTable	dot1dStp	12.8.1
ieee8021SpanningTreeComponentId*		—
ieee8021SpanningTreeProtocolSpecification	dot1dStpProtocolSpecification	—
ieee8021SpanningTreePriority	dot1dStpPriority	12.8.1.1.3 a)
ieee8021SpanningTreeTimeSinceTopologyChange	dot1dStpTimeSinceTopologyChange	12.8.1.1.3 b)
ieee8021SpanningTreeTopChanges	dot1dStpTopChanges	12.8.1.1.3 c)
ieee8021SpanningTreeDesignatedRoot	dot1dStpDesignatedRoot	12.8.1.1.3 e)
ieee8021SpanningTreeRootCost	dot1dStpRootCost	12.8.1.1.3 f)
ieee8021SpanningTreeRootPort	dot1dStpRootPort	12.8.1.1.3 g)
ieee8021SpanningTreeMaxAge	dot1dStpMaxAge	12.8.1.1.3 h)

Table 17-5—IEEE8021-SPANNING-TREE MIB structure and relationship to IETF RFC 4318 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021SpanningTreeHelloTime	dot1dStpHelloTime	12.8.1.1.3 k)
ieee8021SpanningTreeHoldTime	dot1dStpHoldTime	12.8.1.1.3 m)
ieee8021SpanningTreeForwardDelay	dot1dStpForwardDelay	12.8.1.1.3 i)
ieee8021SpanningTreeBridgeMaxAge	dot1dStpBridgeMaxAge	12.8.1.1.3 j)
ieee8021SpanningTreeBridgeHelloTime	dot1dStpBridgeHelloTime	12.8.1.1.3 k)
ieee8021SpanningTreeBridgeForwardDelay	dot1dStpBridgeForwardDelay	12.8.1.1.3 l)
ieee8021SpanningTreeVersion	dot1dStpVersion	12.8.1.1.3 n)
ieee8021SpanningTreeRstpTxHoldCount	dot1dStpTxHoldCount	12.8.1.1.3 m)
ieee8021SpanningTreePortTable	dot1dStpPortTable	12.8.2
ieee8021SpanningTreePortComponentId*	—	—
ieee8021SpanningTreePort*	dot1dStpPort	12.8.2.1.2 a)
ieee8021SpanningTreePortPriority	dot1dStpPortPriority	12.8.2.1.3 c)
ieee8021SpanningTreePortState	dot1dStpPortState	12.8.2.1.3 b)
ieee8021SpanningTreePortEnable	dot1dStpPortEnable	12.8.2.1.3 m)
ieee8021SpanningTreePortPathCost	dot1dStpPortPathCost	12.8.2.1.3 d)
ieee8021SpanningTreePortDesignatedRoot	dot1dStpPortDesignatedRoot	12.8.2.1.3 e)
ieee8021SpanningTreePortDesignatedCost	dot1dStpPortDesignatedCost	12.8.2.1.3 f)
ieee8021SpanningTreePortDesignatedBridge	dot1dStpPortDesignatedBridge	12.8.2.1.3 g)
ieee8021SpanningTreePortDesignatedPort	dot1dStpPortDesignatedPort	12.8.2.1.3 h)
ieee8021SpanningTreePortForwardTransitions	dot1dStpPortForwardTransitions	—
ieee8021SpanningTreeRstpPortProtocolMigration	dot1dStpPortProtocolMigration	12.8.2.5
ieee8021SpanningTreeRstpPortAdminEdgePort	dot1dStpPortAdminEdgePort	12.8.2.1.3 k)
ieee8021SpanningTreeRstpPortOperEdgePort	dot1dStpPortOperEdgePort	12.8.2.1.3 l)
ieee8021SpanningTreeRstpPortAdminPathCost	dot1dStpPortAdminPathCost	12.8.2.1.3 d)
ieee8021SpanningTreePortExtensionTable	AUGMENTS ieee8021SpanningTreePortEntry	12.8.2
ieee8021SpanningTreeRstpPortAutoEdgePort	—	12.8.2.1.3
ieee8021SpanningTreeRstpPortAutoIsolatePort	—	12.8.2.1.3
ieee8021SpanningTreeRstpPortIsolatePort	—	12.8.2.1.3

*This object is an INDEX of the table in which it resides.

The IEEE8021-SPANNING-TREE MIB contains the following object subtrees:

- a) The ieee8021SpanningTree Subtree
This subtree contains the objects that denote the Bridge's state with respect to the Spanning Tree Protocol.
- b) The ieee8021SpanningTreePort Subtree
This subtree contains the objects that denote the Bridge Port's state with respect to the Spanning Tree Protocol.

For compatibility with the original MIB defined in IETF RFC 4318, the IEEE8021-SPANNING-TREE-MIB continues to use disabled, blocking, listening, learning, forwarding, and broken ieee8021SpanningTreePortStates. The learning and forwarding states correspond exactly to the Learning and Forwarding Port States specified in Clause 12 of this standard. Disabled, blocking, listening, and broken all correspond to the Discarding Port State—while those ieee8021SpanningTreePortStates serve to distinguish reasons for discarding frames, the operation of the Forwarding and Learning processes is the same for all of them. The ieee8021SpanningTreePortState broken represents the failure or unavailability of the Port's MAC as indicated by MAC_Operational FALSE; disabled represents exclusion of the Port from the active topology by management setting of the Administrative Port State to Disabled; blocking represents exclusion of the Port from the active topology by the spanning tree algorithm [computing an Alternate or Backup Port Role (17.7 of IEEE Std 802.1D-2004)]; listening represents a Port that the spanning tree algorithm has selected to be part of the active topology (computing a Root Port or Designated Port role) but is temporarily discarding frames to guard against loops or incorrect learning.

The IEEE 802.1D management objects in Table 17-6 have not been included in the STP-MIB module for the indicated reasons.

Table 17-6—Clause 12 objects not in the IEEE8021-SPANNING-TREE MIB

IEEE 802.1D object	Disposition
SpanningTreeProtocol	
.BridgeIdentifier	Combination of ieee8021SpanningTreePriority and ieee8021BridgeBaseBridgeAddress
SpanningTreeProtocolPort	
.Uptime	Same as ifLastChange (IF-MIB)
.PortIdentifier	Combination of ieee8021SpanningTreePort and ieee8021SpanningTreePortPriority
.DiscardLackOfBuffers	Redundant

17.2.4 Structure of the IEEE8021-Q-BRIDGE MIB

A Q-BRIDGE MIB originally appeared in IETF RFC 4363. While many tables and objects are similar, it has been reindexed and rerooted with its inclusion in this subclause.

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. Where appropriate, the corresponding IETF RFC 4363 object name and IEEE Std 802.1Q management reference is also included.

Unlike IEEE Std 802.1D, 1998 Edition, this standard does not define exact syntax for a set of managed objects. The following cross-references indicate only the subclause numbering of the descriptions of management operations from Clause 12.

Table 17-7 indicates the structure of the IEEE8021-Q-BRIDGE MIB module as well as showing its relationship to the IETF MIB that is being obsoleted.

Table 17-7—IEEE8021-QBRIDGE MIB structure and relationship to IETF RFC 4363 and this standard

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021QBridgeTable	dot1qBase	12.4
ieee8021QBridgeComponentId [*]	—	—
ieee8021QBridgeVlanVersionNumber	dot1qVlanVersionNumber	12.10.1.1
ieee8021QBridgeMaxVlanId	dot1qMaxVlanId	9.6
ieee8021QBridgeMaxSupportedVlans	dot1qMaxSupportedVlans	12.10.1.1
ieee8021QBridgeNumVlans	dot1qNumVlans	12.7.1.1
ieee8021QBridgeMvrpEnabledStatus	dot1qGvrpStatus	—
ieee8021QBridgeCVlanPortTable		
ieee8021QBridgeCVlanPortComponentId [*]		
ieee8021QBridgeCVlanPortNumber		
ieee8021QBridgeCVlanPortRowStatus		
ieee8021QBridgeFdbTable	dot1qFdbTable	12.7.1
ieee8021QBridgeFdbComponentId [*]	—	—
ieee8021QBridgeFdbId [*]	dot1qFdbId	—
ieee8021QBridgeFdbDynamicCount	dot1qFdbDynamicCount	12.7.1.1.3
ieee8021QBridgeFdbLearnedEntryDiscards		
ieee8021QBridgeFdbAgingTime		
ieee8021QBridgeTpFdbTable	dot1qTpFdbTable	12.7.1
ieee8021QBridgeFdbComponentId [*]	—	—
ieee8021QBridgeFdbId [*]	—	—
ieee8021QBridgeTpFdbAddress [*]	dot1qTpFdbAddress	—
ieee8021QBridgeTpFdbPort	dot1qTpFdbPort	—

Table 17-7—IEEE8021-QBRIDGE MIB structure and relationship to IETF RFC 4363 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021QBridgeTpFdbStatus	dot1qTpFdbStatus	—
ieee8021QBridgeTpGroupTable	dot1qTpGroupTable	12.7.4
ieee8021QBridgeVlanCurrentComponentId*	—	—
ieee8021QBridgeVlanIndex*	—	—
ieee8021QBridgeTpGroupAddress*	dot1qTpGroupAddress	—
ieee8021QBridgeTpGroupEgressPorts	dot1qTpGroupEgressPorts	—
ieee8021QBridgeTpGroupLearnt	dot1qTpGroupLearnt	—
ieee8021QBridgeForwardAllTable	dot1qForwardAllTable	12.7.2 12.7.7
ieee8021QBridgeForwardAllComponentId*	—	—
ieee8021QBridgeVlanIndex*	—	—
ieee8021QBridgeForwardAllPorts	dot1qForwardAllPorts	—
ieee8021QBridgeForwardAllStaticPorts	dot1qForwardAllStaticPorts	—
ieee8021QBridgeForwardAllForbiddenPorts	dot1qForwardAllForbiddenPorts	—
ieee8021QBridgeForwardUnregisteredTable	dot1qForwardUnregisteredTable	12.7.2 12.7.7
ieee8021QBridgeForwardUnregisteredComponentId*	—	—
ieee8021QBridgeVlanIndex*	—	—
ieee8021QBridgeForwardUnregisteredPorts	dot1qForwardUnregisteredPorts	—
ieee8021QBridgeForwardUnregisteredStaticPorts	dot1qForwardUnregisteredStaticPorts	—
ieee8021QBridgeForwardUnregisteredForbiddenPorts	dot1qForwardUnregisteredForbiddenPorts	—
ieee8021QBridgeStaticUnicastTable	dot1qStaticUnicastTable	12.7.7, 8.8.1
ieee8021QBridgeFdbComponentId*	—	—
ieee8021QBridgeFdbId*	—	—
ieee8021QBridgeStaticUnicastAddress*	dot1qStaticUnicastAddress	—
ieee8021QBridgeStaticUnicastReceivePort*	dot1qStaticUnicastReceivePort	—
ieee8021QBridgeStaticUnicastStaticEgressPorts	dot1qStaticUnicastAllowedToGoTo	8.5

Table 17-7—IEEE8021-QBRIDGE MIB structure and relationship to IETF RFC 4363 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021QBridgeStaticUnicastForbiddenEgressPorts	dot1qStaticUnicastAllowedToGoTo	8.5
ieee8021QBridgeStaticUnicastStorageType		
ieee8021QBridgeStaticUnicastRowStatus	dot1qStaticUnicastStatus	—
ieee8021QBridgeStaticMulticastTable	dot1qStaticMulticastTable	12.7.7, 8.8.1
ieee8021QBridgeVlanCurrentComponentId*	—	—
ieee8021QBridgeVlanIndex*	—	—
ieee8021QBridgeStaticMulticastAddress*	dot1qStaticMulticastAddress	—
ieee8021QBridgeStaticMulticastReceivePort*	dot1qStaticMulticastReceivePort	—
ieee8021QBridgeStaticMulticastStaticEgressPorts	dot1qStaticMulticastStaticEgressPorts	—
ieee8021QBridgeStaticMulticastForbiddenEgressPorts	dot1qStaticMulticastForbiddenEgressPorts	—
ieee8021QBridgeStaticMulticastStorageType		—
ieee8021QBridgeStaticMulticastRowStatus	dot1qStaticMulticastStatus	—
ieee8021QBridgeVlan	dot1qVlan	—
ieee8021QBridgeVlanNumDeletes	dot1qVlanNumDeletes	—
ieee8021QBridgeVlanCurrentTable	dot1qVlanCurrentTable	12.10.2
ieee8021QBridgeVlanCurrentComponentId*	—	—
ieee8021QBridgeVlanTimeMark*	dot1qVlanTimeMark	—
ieee8021QBridgeVlanIndex*	dot1qVlanIndex	—
ieee8021QBridgeVlanFdbId	dot1qVlanFdbId	—
ieee8021QBridgeVlanCurrentEgressPorts	dot1qVlanCurrentEgressPorts	12.10.2.1
ieee8021QBridgeVlanCurrentUntaggedPorts	dot1qVlanCurrentUntaggedPorts	12.10.2.1
ieee8021QBridgeVlanStatus	dot1qVlanStatus	—
ieee8021QBridgeVlanCreationTime	dot1qVlanCreationTime	—
ieee8021QBridgeVlanStaticTable	dot1qVlanStaticTable	12.7.5
ieee8021QBridgeVlanStaticComponentId*	—	—

Table 17-7—IEEE8021-QBRIDGE MIB structure and relationship to IETF RFC 4363 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021QBridgeVlanStaticVlanIndex*	—	—
ieee8021QBridgeVlanStaticName	dot1qVlanStaticName	12.10.2.1
ieee8021QBridgeVlanStaticEgressPorts	dot1qVlanStaticEgressPorts	12.7.7.3, 11.2.3.2.3
ieee8021QBridgeVlanForbiddenEgressPorts	dot1qVlanForbiddenEgressPorts	12.7.7.3, 11.2.3.2.3
ieee8021QBridgeVlanStaticUntaggedPorts	dot1qVlanStaticUntaggedPorts	12.10.2.1
ieee8021QBridgeVlanStaticRowStatus	dot1qVlanStaticRowStatus	—
ieee8021QBridgeNextFreeLocalVlanTable	dot1qVlan	—
ieee8021QBridgeNextFreeLocalVlanComponentId*	—	—
ieee8021QBridgeNextFreeLocalVlanIndex	dot1qNextFreeLocalVlanIndex	—
ieee8021QBridgePortVlanTable	dot1qPortVlanTable	12.10.1
(AUGMENTS ieee8021BridgeBasePortEntry)	(AUGMENTS dot1dBasePortEntry)	—
ieee8021QBridgePvid	dot1qPvid	12.10.1.1
ieee8021QBridgePortAcceptableFrameTypes	dot1qPortAcceptableFrameTypes	12.10.1.3
ieee8021QBridgePortIngressFiltering	dot1qPortIngressFiltering	12.10.1.4
ieee8021QBridgePortmvrpEnabledStatus	dot1qPortGvrpStatus	—
ieee8021QBridgePortmvrpFailedRegistrations	dot1qPortGvrpFailedRegistrations	—
ieee8021QBridgePortmvrpLastPduOrigin	dot1qPortGvrpLastPduOrigin	—
ieee8021QBridgePortRestrictedVlanRegistration	dot1qPortRestrictedVlanRegistration	11.2.3.2.3, 12.10.1.7
ieee8021QBridgePortVlanStatisticsTable	dot1qPortVlanStatisticsTable	12.1.3
ieee8021BridgeBasePortComponentId*	—	—
ieee8021BridgeBasePort*	dot1dBasePort	—
ieee8021QBridgeVlanIndex*	dot1qVlanIndex	—
ieee8021QBridgeTpVlanPortInFrames	dot1qTpVlanPortInFrames dot1qTpVlanPortHCInFrames	12.6.1.1.3(a)
ieee8021QBridgeTpVlanPortOutFrames	dot1qTpVlanPortOutFrames dot1qTpVlanPortHCOutFrames	12.6.1.1.3(d)

Table 17-7—IEEE8021-QBRIDGE MIB structure and relationship to IETF RFC 4363 and this standard (continued)

IEEE MIB table/object	IETF MIB table/object	Reference
ieee8021QBridgeTpVlanPortInDiscards	dot1qTpVlanPortInDiscards dot1qTpVlanPortHCInDiscards	12.6.1.1.3
ieee8021QBridgeLearningConstraintsTable	dot1qLearningConstraintsTable	12.10.3.1
ieee8021QBridgeLearningConstraintsComponentId [*]		—
ieee8021QBridgeConstraintVlan [*]	dot1qConstraintVlan	—
ieee8021QBridgeConstraintSet [*]	dot1qConstraintSet	—
ieee8021QBridgeConstraintType	dot1qConstraintType	—
ieee8021QBridgeConstraintStatus	dot1qConstraintStatus	—
ieee8021QBridgeLearningConstraintDefaultsTable	dot1qVlan	12.10.3.1
ieee8021QBridgeLearningConstraintDefaultsComponentId [*]	—	—
ieee8021QBridgeLearningConstraintDefaultsSet	dot1qConstraintSetDefault	—
ieee8021QBridgeLearningConstraintDefaultsType	dot1qConstraintTypeDefault	—
ieee8021QBridgeProtocolGroupTable	dot1vProtocolGroupTable	12.10.1
ieee8021QBridgeProtocolGroupComponentId [*]	—	—
ieee8021QBridgeProtocolTemplateFrameType [*]	dot1vProtocolTemplateFrameType	12.10.1.7
ieee8021QBridgeProtocolTemplateProtocolValue [*]	dot1vProtocolTemplateProtocolValue	12.10.1.7
ieee8021QBridgeProtocolGroupId	dot1vProtocolGroupId	12.10.1.7
ieee8021QBridgeProtocolGroupRowStatus	dot1vProtocolGroupRowStatus	—
ieee8021QBridgeProtocolPortTable	dot1vProtocolPortTable	12.10.1
ieee8021BridgeBasePortComponentId [*]	—	—
ieee8021BridgeBasePort [*]	dot1dBasePort	—
ieee8021QBridgeProtocolPortGroupId [*]	dot1vProtocolPortGroupId	12.10.1.7
ieee8021QBridgeProtocolPortGroupVid	dot1vProtocolPortGroupVid	12.10.1.7
ieee8021QBridgeProtocolPortRowStatus	dot1vProtocolPortRowStatus	—

*This object is an INDEX of the table in which it resides.

The IEEE8021-Q-BRIDGE MIB contains the following object subtrees:

- a) The ieee8021QBridgeBase Subtree
This subtree contains the objects that are applicable to all Bridges implementing IEEE Std 802.1Q virtual LANs.
- b) The ieee8021QBridgeTp Subtree
This subtree contains objects that control the operation and report the status of transparent bridging. This includes management of the dynamic Filtering Databases for both unicast and multicast forwarding. This subtree will be implemented by all Bridges that perform destination-address filtering.
- c) The ieee8021QBridgeStatic Subtree
This subtree contains objects that control static configuration information for transparent bridging. This includes management of the static entries in the Filtering Databases for both unicast and multicast forwarding.
- d) The ieee8021QBridgeVlan Subtree
This subtree contains objects that control configuration and report status of the Virtual LANs known to a Bridge. This includes management of the statically configured VIDs as well as reporting VIDs discovered by other means [e.g., MRP VLAN Registration Protocol (MVRP)]. It also controls configuration and reports status of per-Port objects relating to VIDs and reports traffic statistics. It also provides for management of the VLAN Learning Constraints.
- e) The ieee8021QBridgeProtocol Subtree
This subtree contains objects that control configuration and report status of the mappings from Protocol Templates to Protocol Group Identifiers used for Port-and-Protocol-based VLAN Classification.

The Clause 12 management objects in Table 17-8 have not been included in the IEEE8021-Q-BRIDGE MIB for the indicated reasons. Note that the assumption is made the base modules SNMPv2-MIB and IF-MIB will be implemented as a default on all Bridges.

Table 17-8—Clause 12 management not in IEEE8021-Q-BRIDGE MIB

Clause 12 operation	Disposition
Reset Bridge (12.4.1.4)	Not appropriate for this MIB module
Reset VLAN Bridge (12.10.1.5)	Not appropriate for this MIB module
Read permanent database (12.7.6.1)	Count rows
Permanent database size	Count rows
Number of static filtering entries	Count rows in ieee8021QBridgeStaticUnicastTable + ieee8021QBridgeStaticMulticastTable
Number of static VLAN registration entries	Count rows in ieee8021QBridgeVlanStaticTable
Read filtering entry range (12.7.7.4)	Use GetNext operation
Read filtering database (12.7.1.1)	Count rows
Filtering database size	Count rows
Number of dynamic group address entries	Count rows applicable to each FDB in ieee8021QBridgeTpGroupTable

17.2.5 Structure of the IEEE8021-PB MIB

The IEEE8021-PB MIB provides objects to configure and manage provider bridges.

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. Where appropriate, the corresponding Clause 12 management reference is also included.

Table 17-9 indicates the structure of the IEEE8021-PB MIB module.

Table 17-9—IEEE8021-PB MIB structure and relationship to this standard

IEEE MIB table	IEEE MIB object	Reference
ieee8021PbVidTranslationTable		12.13.2.1, 12.13.2.2
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbVidTranslationLocalVid*	—
	ieee8021PbVidTranslationRelayVid	—
	ieee8021PbVidTranslationRowStatus	—
ieee8021PbCVidRegistrationTable		12.13.3.1, 12.13.3.2
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbCVidRegistrationCVid*	—
	ieee8021PbCVidRegistrationSVid	—
	ieee8021PbCVidRegistrationUntaggedPep	—
	ieee8021PbCVidRegistrationUntaggedCep	—
	ieee8021PbCVidRegistrationRowStatus	—
ieee8021PbEdgePortTable		12.13.3.3, 12.13.3.4
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbEdgePortSVid*	—
	ieee8021PbEdgePortPVID	—
	ieee8021PbEdgePortDefaultUserPriority	—
	ieee8021PbEdgePortAcceptableFrameTypes	—

Table 17-9—IEEE8021-PB MIB structure and relationship to this standard (continued)

IEEE MIB table	IEEE MIB object	Reference
	ieee8021PbEdgePortEnableIngressFiltering	—
ieee8021PbServicePriorityRegenerationTable		12.13.3.5, 12.13.3.6
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbServicePriorityRegenerationSVid*	—
	ieee8021PbServicePriorityRegenerationReceivedPriority*	—
	ieee8021PbServicePriorityRegenerationRegeneratedPriority	—
ieee8021PbCnpTable		12.13.3
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbCnpCComponentId	—
	ieee8021PbCnpSVid	—
	ieee8021PbCnpRowStatus	—
ieee8021PbPnpTable		12.13.3
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbPnpRowStatus	—
ieee8021PbCepTable		12.13.3
	ieee8021BridgeBasePortComponentId*	—
	ieee8021BridgeBasePort*	—
	ieee8021PbCepCComponentId	—
	ieee8021PbCepCepPortNumber	—
	ieee8021PbCepRowStatus	—

*This object is an INDEX of the table in which it resides.

17.2.6 Structure of the IEEE8021-MSTP MIB

The IEEE8021-MSTP MIB provides objects to configure and manage multiple spanning trees.

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. Where appropriate, the corresponding Clause 12 management reference is also included.

Table 17-10 indicates the structure of the IEEE8021-MSTP MIB module.

Table 17-10—IEEE8021-MSTP MIB structure and relationship to this standard

IEEE MIB table	IEEE MIB object	Reference
ieee8021MstpCistTable		12.8.1.1, 12.8.1.3
	ieee8021MstpCistComponentId*	—
	ieee8021MstpCistBridgeIdentifier	—
	ieee8021MstpCistTopologyChange	—
	ieee8021MstpCistRegionalRootIdentifier	13.24.2
	ieee8021MstpCistPathCost	—
	ieee8021MstpCistMaxHops	13.24.3
ieee8021MstpTable		12.8.1.2, 12.8.1.4, 12.12.3.2, 12.12.1
	ieee8021MstpComponentId*	—
	ieee8021MstpId*	—
	ieee8021MstpBridgeId	13.24.1
	ieee8021MstpTimeSinceTopologyChange	13.21
	ieee8021MstpTopologyChanges	13.21
	ieee8021MstpTopologyChange	—
	ieee8021MstpDesignatedRoot	13.24.2
	ieee8021MstpRootPathCost	13.24.2
	ieee8021MstpRootPort	13.24.7
	ieee8021MstpBridgePriority	13.24.2
	ieee8021MstpVids0	
	ieee8021MstpVids1	
	ieee8021MstpVids2	
	ieee8021MstpVids3	

Table 17-10—IEEE8021-MSTP MIB structure and relationship to this standard (continued)

IEEE MIB table	IEEE MIB object	Reference
	ieee8021MstpRowStatus	—
ieee8021MstpCistPortTable		12.8.2.1, 12.8.2.3, 12.8.2.5
	ieee8021MstpCistPortComponentId*	—
	ieee8021MstpCistPortNum*	—
	ieee8021MstpCistPortUptime	—
	ieee8021MstpCistPortAdminPathCost	IEEE 802.1D ^a 13.22 p), 17.13.1
	ieee8021MstpCistPortDesignatedRoot	13.25.7
	ieee8021MstpCistPortTopologyChangeAck	IEEE 802.1D 17.19.41
	ieee8021MstpCistPortAdminHelloTime	12.8.2.1
	ieee8021MstpCistPortAdminEdgePort	IEEE 802.1D 17.13.1
	ieee8021MstpCistPortOperEdgePort	IEEE 802.1D 17.19.17
	ieee8021MstpCistPortOperHelloTime	12.8.2.1
	ieee8021MstpCistPortMacEnabled	12.8.2.1.3 m)
	ieee8021MstpCistPortMacOperational	12.8.2.1.3 n)
	ieee8021MstpCistPortRestrictedRole	13.25.49
	ieee8021MstpCistPortRestrictedTcn	13.25.50
	ieee8021MstpCistPortRole	—
	ieee8021MstpCistPortDisputed	IEEE 802.1D 13.24 u) and 17.19.6
	ieee8021MstpCistPortCistRegionalRootId	13.9 c), 13.10, 13.25.33
	ieee8021MstpCistPortCistPathCost	13.9 d), 13.10, 13.25.33
	ieee8021MstpCistPortProtocolMigration	IEEE 802.1D 13.24 u) and 17.19.6
	ieee8021MstpCistPortEnableBPDUrx	13.25.16
	ieee8021MstpCistPortEnableBPDUtx	13.25.17
	ieee8021MstpCistPortPseudoRootId	13.25.18
	ieee8021MstpCistPortIsL2Gp	13.25.16
ieee8021MstpPortTable		12.8.2.2, 12.8.2.4
	ieee8021MstpPortComponentId*	—
	ieee8021MstpPortMstpId*	—

Table 17-10—IEEE8021-MSTP MIB structure and relationship to this standard (continued)

IEEE MIB table	IEEE MIB object	Reference
	ieee8021MstpPortNum*	—
	ieee8021MstpPortUptime	—
	ieee8021MstpPortState	IEEE 802.1D 13.35 and 17.10
	ieee8021MstpPortPriority	13.25.33
	ieee8021MstpPortPathCost	13.25.12
	ieee8021MstpPortDesignatedRoot	13.25.7
	ieee8021MstpPortDesignatedCost	13.25.7
	ieee8021MstpPortDesignatedBridge	13.25.7
	ieee8021MstpPortDesignatedPort	13.25.7
	ieee8021MstpPortRole	—
	ieee8021MstpPortDisputed	IEEE 802.1D 13.24 u) and 17.19.6
ieee8021MstpFidToMstiTable		12.12.2.2
	ieee8021MstpFidToMstiComponentId*	—
	ieee8021MstpFidToMstiFid*	—
	ieee8021MstpFidToMstiMstpId	—
ieee8021MstpVlanTable		12.12.3.1
	ieee8021MstpVlanComponentId*	—
	ieee8021MstpVlanId*	—
	ieee8021MstpVlanMstpId	—
ieee8021MstpConfigIdTable		12.12.3.3, 12.12.3.4
	ieee8021MstpConfigIdComponentId*	—
	ieee8021MstpConfigIdFormatSelector	13.7:1
	ieee8021MstpConfigurationName	13.7:2
	ieee8021MstpRevisionLevel	13.7:3

Table 17-10—IEEE8021-MSTP MIB structure and relationship to this standard (continued)

IEEE MIB table	IEEE MIB object	Reference
	ieee8021MstpConfigurationDigest	13.7:4
ieee8021MstpCistPortExtensionTable AUGMENTS Ieee8021MstpCistPortEntry		12.8.2
	ieee8021MstpCistPortAutoEdgePort	12.8.2.1.3
	ieee8021MstpCistPortAutoIsolatePort	12.8.2.1.3

*This object is an INDEX of the table in which it resides.

^aReferences in this table to *IEEE 802.1D* are to IEEE Std 802.1D-2004.

17.2.7 Structure of the IEEE8021-CFM MMIB

Subclause 12.14 of this document defines the information model associated with this standard in a protocol independent manner. Table 17-11 and Table 17-12 describe the relationship between the SMIv2 objects defined in the MIB module in 17.4.9, the variables defined in Clause 20, and the protocol-independent objects defined in 12.14.

Note that there are actually two MIB modules—IEEE8021-CFM and IEEE8021-CFM-V2. The former contains all original and current tables as well as deprecated tables. The latter contains reindexed tables (to support PBB per Clause 26) and a complete conformance clause that lists all the current tables. Table 17-11 lists only the current tables (i.e., the deprecated tables are not listed) in IEEE8021-CFM MIB module and Table 17-12 lists the reindexed tables in IEEE8021-CFM-V2 MIB module (that replace the deprecated tables of IEEE8021-CFM).

Table 17-11—IEEE8021-CFM MIB correspondence between variables, managed objects, and MIB objects

Variable	IEEE MIB table/object	Reference
Maintenance Domain managed object (12.14.5)	dot1agCfmMdTable	
	dot1agCfmMdIndex*	12.14.1.1.3 a2)
	dot1agCfmMdFormat, dot1agCfmMdName	12.14.1.2.2 a)
mdLevel (20.7.1)	dot1agCfmMdMdLevel	12.14.5.1.3 b)
	dot1agCfmMdMhfCreation	12.14.5.1.3 c)
	dot1agCfmMdMhfIdPermission	12.14.5.1.3 d)
	controlled by IETF RFC 3413	12.14.5.1.3 e)
	dot1agCfmMdIndex	12.14.5.1.3 f)
Maintenance Association managed object (12.14.6)	dot1agCfmMaNetTable, dot1agCfmMaMepListTable	
	dot1agCfmMaIndex*	12.14.6.1.2 a)

Table 17-11—IEEE8021-CFM MIB correspondence between variables, managed objects, and MIB objects (*continued*)

Variable	IEEE MIB table/object	Reference
	dot1agCfmMaNetFormat, dot1agCfmMaNetName	12.14.5.3.2 b)
	dot1agCfmVlanVid, dot1agCfmMaCompNumberOfVids	12.14.6.1.3 b)
	dot1agCfmMaCompMhfCreation	12.14.6.1.3 c)
	dot1agCfmMaCompIdPermission	12.14.6.1.3 d)
CCMinterval (20.8.1)	dot1agCfmMaNetCcmInterval	12.14.6.1.3 e)
	controlled by IETF RFC 3413	12.14.6.1.3 f)
	dot1agCfmMaMepListIdentifier	12.14.6.1.3 g)
MEP managed object (12.14.7)	dot1agCfmMepTable	
	dot1agCfmMdIndex*, dot1agCfmMaIndex*, dot1agCfmMepIdentifier*	12.14.7.1.2 a)
	dot1agCfmMepIfIndex	12.14.7.1.3 b)
	dot1agCfmMepDirection	12.14.7.1.3 c)
	dot1agCfmMepPrimaryVid	12.14.7.1.3 d)
MEPactive (20.9.1)	dot1agCfmMepActive	12.14.7.1.3 e)
20.37	dot1agCfmMepFngState	12.14.7.1.3 f)
CCIenabled (20.10.1)	dot1agCfmMepCciEnabled	12.14.7.1.3 g)
	dot1agCfmMepCcmLtmPriority	12.14.7.1.3 h)
	dot1agCfmMepMacAddress	12.14.7.1.3 i)
	controlled by IETF RFC 3413	12.14.7.1.3 j)
lowestAlarmPri (20.9.5)	dot1agCfmMepLowPrDef	12.14.7.1.3 k)
fngAlarmTime (20.35.3)	dot1agCfmMepFngAlarmTime	12.14.7.1.3 l)
fngResetTime (20.35.4)	dot1agCfmMepFngResetTime	12.14.7.1.3 m)
highestDefect (20.35.9)	dot1agCfmMepHighestPrDefect	12.14.7.1.3 n)
someRDIdefect (20.35.7)	dot1agCfmMepDefects	12.14.7.1.3 o)
someMACstatusDefect (20.35.6)		12.14.7.1.3 p)
someRMEPCCMdefect (20.35.5)		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)	dot1agCfmMepErrorCcmLastFailure	12.14.7.1.3 t)
xconCCMlastFailure (20.23.2)	dot1agCfmMepXconCcmLastFailure	12.14.7.1.3 u)
CCMsequenceErrors (20.16.12)	dot1agCfmMepCcmSequenceErrors	12.14.7.1.3 v)

Table 17-11—IEEE8021-CFM MIB correspondence between variables, managed objects, and MIB objects (*continued*)

Variable	IEEE MIB table/object	Reference
CCIsentCCMs (20.10.2)	dot1agCfmMepCciSentCcms	12.14.7.1.3 w)
nextLBMtransID (20.30.2)	dot1agCfmMepNextLbmTransId	12.14.7.1.3 x)
	dot1agCfmMepLbrIn	12.14.7.1.3 y)
	dot1agCfmMepLbrInOutOfOrder	12.14.7.1.3 z)
	dot1agCfmMepLbrBadMsdu	12.14.7.1.3 aa)
nextLTMtransID (20.41.1)	dot1agCfmMepLtmNextSeqNumber	12.14.7.1.3 ab)
	dot1agCfmMepUnexpLtrIn	12.14.7.1.3 ac)
	dot1agCfmMepLbrOut	12.14.7.1.3 ad)
	dot1agCfmMepPbbTeCanReportPbbTePresence	12.14.7.1.3 af)
	dot1agCfmMepPbbTeTrafficMismatchDefect	12.14.7.1.3 ah)
	dot1agCfmMepPbbTeLbmReverseVid	12.14.7.3.2 f)
	dot1agCfmMepPbbTeLtmReverseVid	12.14.7.4.2e)
	dot1agCfmMepPbbTeMismatchAlarm	12.14.7.1.3 ag)
	dot1agCfmMepPbbTeMismatchDefect	12.14.7.1.3 ai)
	dot1agCfmMepPbbTeMismatchSinceReset	12.14.7.1.3 aj)
Transmit Loopback Messages (12.14.7.3)	dot1agCfmMepTable , dot1agCfmMepTransmitLbmStatus	
	dot1agCfmMdIndex*, dot1agCfmMaIndex*, dot1agCfmMepIdentifier*	12.14.7.3.2 a)
	dot1agCfmMepTransmitLbmDestMacAddress, dot1agCfmMepTransmitLbmDestMepId, dot1agCfmMepTransmitLbmDestIsMepId	12.14.7.3.2 b)
	dot1agCfmMepTransmitLbmMessages	12.14.7.3.2 c)
	dot1agCfmMepTransmitLbmDataTlv	12.14.7.3.2 d)
	dot1agCfmMepTransmitLbmVlanPriority, dot1agCfmMepTransmitLbmVlanDropEnable	12.14.7.3.2 e)
	dot1agCfmMepTransmitLbmResultOK	12.14.7.3.3 a)
	dot1agCfmMepTransmitLbmSeqNumber	12.14.7.3.3 b)
Transmit Linktrace Message (12.14.7.4)	dot1agCfmMepTable	
	dot1agCfmMdIndex*, dot1agCfmMaIndex*, dot1agCfmMepIdentifier*	12.14.7.4.2 a)
	dot1agCfmMepTransmitLtmFlags	12.14.7.4.2 b)

Table 17-11—IEEE8021-CFM MIB correspondence between variables, managed objects, and MIB objects (*continued*)

Variable	IEEE MIB table/object	Reference
	dot1agCfmMepTransmitLtmTargetMacAddress dot1agCfmMepTransmitLtmTargetMepId, dot1agCfmMepTransmitLtmTargetIsMepId	12.14.7.4.2 c)
	dot1agCfmMepTransmitLtmTtl	12.14.7.4.2 d)
	dot1agCfmMepTransmitLtmResult	12.14.7.4.3 a)
	dot1agCfmMepTransmitLtmSeqNumber	12.14.7.4.3 b)
	dot1agCfmMepTransmitLtmEgressIdentifier	12.14.7.4.3 c)
Read Linktrace Reply (12.14.7.5)	dot1agCfmLtrTable	
	dot1agCfmMdIndex*, dot1agCfmMaIndex*, dot1agCfmMepIdentifier*	12.14.7.5.2 a)
	dot1agCfmLtrSeqNumber*	12.14.7.5.2 b)
	dot1agCfmLtrReceiveOrder*	12.14.7.5.2 c)
ltrReplyTTL (20.41.2.2)	dot1agCfmLtrTtl	12.14.7.5.3 b)
ltrFlags (20.41.2.1)	dot1agCfmLtrForwarded	12.14.7.5.3 c)
	dot1agCfmLtrTerminalMep	12.14.7.5.3 d)
ltrLastEgressId (20.41.2.3)	dot1agCfmLtrLastEgressIdentifier	12.14.7.5.3 e)
ltrNextEgressId (20.41.2.4)	dot1agCfmLtrNextEgressIdentifier	12.14.7.5.3 f)
ltrRelayAction (20.41.2.5)	dot1agCfmLtrRelay	12.14.7.5.3 g)
ltrIngressAction (20.41.2.6)	dot1agCfmLtrIngress	12.14.7.5.3 k)
ltrIngressAddress (20.41.2.7)	dot1agCfmLtrIngressMac	12.14.7.5.3 l)
ltrIngressPortIdSubtype (20.41.2.8)	dot1agCfmLtrIngressPortIdSubtype	12.14.7.5.3 m)
ltrIngressPortId (20.41.2.9)	dot1agCfmLtrIngressPortId	12.14.7.5.3 n)
ltrEgressAction (20.41.2.10)	dot1agCfmLtrEgress	12.14.7.5.3 o)
ltrEgressAddress (20.41.2.11)	dot1agCfmLtrEgressMac	12.14.7.5.3 p)
ltrEgressPortIdSubtype (20.41.2.12)	dot1agCfmLtrEgressPortIdSubtype	12.14.7.5.3 q)
ltrEgressPortId (20.41.2.13)	dot1agCfmLtrEgressPortId	12.14.7.5.3 r)
ltrOrgSpecTlv (20.41.2.15)	dot1agCfmLtrOrganizationSpecificTlv	12.14.7.5.3 s)
ltrSenderIdTlv (20.41.2.14)	dot1agCfmLtrChassisIdSubtype	12.14.7.5.3 h)
	dot1agCfmLtrChassisId	12.14.7.5.3 i)
	dot1agCfmLtrManAddressDomain, dot1agCfmLtrManAddress	12.14.7.5.3 j)

Table 17-11—IEEE8021-CFM MIB correspondence between variables, managed objects, and MIB objects (continued)

Variable	IEEE MIB table/object	Reference
Read MEP Database (12.14.7.6)	dot1agCfmMepDbTable	
	dot1agCfmMdIndex*, dot1agCfmMaIndex*, dot1agCfmMepIdentifier*	12.14.7.6.2 a)
	dot1agCfmMepDbRMepIdentifier*	12.14.7.6.2 b)
20.20	dot1agCfmMepDbRMepState	12.14.7.6.3 b)
	dot1agCfmMepDbRMepFailedOkTime	12.14.7.6.3 c)
rMEPmacAddress (20.19.7)	dot1agCfmMepDbMacAddress	12.14.7.6.3 d)
rMEPlastRDI (20.19.2)	dot1agCfmMepDbRdi	12.14.7.6.3 e)
rMEPlastPortState (20.19.3)	dot1agCfmMepDbPortStatusTlv	12.14.7.6.3 f)
rMEPlastInterfaceStatus (20.19.4)	dot1agCfmMepDbInterfaceStatusTlv	12.14.7.6.3 g)
rMEPlastSenderId (20.19.5)	dot1agCfmMepDbChassisIdSubtype dot1agCfmMepDbChassisId dot1agCfmMepDbManAddressDomain, dot1agCfmMepDbManAddress	12.14.7.6.3 h)
	dot1agCfmMepDbRMepIsActive	12.14.7.1.3 ae)
Transmit MEP Fault Alarm (12.14.7.7)	dot1agCfmFaultAlarm	
	dot1agCfmMepHighestPrDefect	12.14.7.7.2 b), 12.14.7.7.2 c)

*This object is an INDEX of the table in which it resides.

Table 17-12—IEEE8021-CFM-V2 correspondence between variables, managed objects, and MIB objects

Variable	IEEE MIB table/object	Reference
CFM Stack managed object (12.14.2)	ieee8021CfmStackTable	
	ieee8021CfmStackIfIndex*	12.14.2.1.2 a)
	ieee8021CfmStackServiceSelectorType*	12.14.2.1.2 d)
	ieee8021CfmStackServiceSelectorOrNone*	12.14.2.1.2 d)
	ieee8021CfmStackMdLevel*	12.14.2.1.2 b)
	ieee8021CfmStackDirection*	12.14.2.1.2 c)
	ieee8021CfmStackMdIndex	12.14.2.1.3 b)
	ieee8021CfmStackMaIndex	12.14.2.1.2 c)

Table 17-12—IEEE8021-CFM-V2 correspondence between variables, managed objects, and MIB objects (continued)

Variable	IEEE MIB table/object	Reference
	ieee8021CfmStackMepId	12.14.2.1.3 d)
	ieee8021CfmStackMacAddress	12.14.2.1.3 e)
Default MD Level managed object (12.14.3)	ieee8021CfmDefaultMdTable	
	ieee8021CfmDefaultMdComponentId* ieee8021CfmDefaultMdPrimarySelectorType* ieee8021CfmDefaultMdPrimarySelector*	12.14.3.1.3 a), 12.14.3.2.2 a)
	ieee8021CfmDefaultMdStatus	12.14.3.1.3 b)
	ieee8021CfmDefaultMdDefLevel, ieee8021CfmDefaultMdLevel	12.14.3.1.3 c), 12.14.3.2.2 b)
	ieee8021CfmDefaultMdDefMhfCreation, ieee8021CfmDefaultMdMhfCreation	12.14.3.1.3 d)
	ieee8021CfmDefaultMdDefIdPermission, ieee8021CfmDefaultMdIdPermission	12.14.3.1.3 e)
Configuration Error List managed object (12.14.4)	ieee8021CfmConfigErrorListTable	
	ieee8021CfmConfigErrorListSelectorType* ieee8021CfmConfigErrorListSelector*	12.14.4.1.2 a)
	ieee8021CfmConfigErrorListIfIndex*	12.14.4.1.2 b)
	ieee8021CfmConfigErrorListErrorType	12.14.4.1.3 b)
Maintenance Association managed object (12.14.6)	ieee8021CfmVlanTable	12.14.3.1.3 a), 12.14.3.2.2 a), 12.14.5.3.2 c), 12.14.6.1.3 b)
	ieee8021CfmVlanComponentId*	—
	ieee8021CfmVlanSelector*	—
	ieee8021CfmVlanRowStatus	—
Maintenance Association managed object (12.14.6)	ieee8021CfmMaCompTable	
	ieee8021CfmMaComponentId* dot1agCfmMdIndex* dot1agCfmMaIndex*	12.14.6.1.2 a)
	ieee8021CfmMaCompPrimarySelectorType, ieee8021CfmMaCompPrimarySelectorOrNone	
	ieee8021CfmMaNetFormat, ieee8021CfmMaNetName	12.14.5.3.2 b)
	ieee8021CfmMaCompNumberOfVids	12.14.6.1.3 b)

Table 17-12—IEEE8021-CFM-V2 correspondence between variables, managed objects, and MIB objects (continued)

Variable	IEEE MIB table/object	Reference
	ieee8021CfmMaCompMhfCreation	12.14.6.1.3 c)
	ieee8021CfmMaCompIdPermission	12.14.6.1.3 d)
CCMinterval (20.8.1)	ieee8021CfmMaNetCcmInterval	12.14.6.1.3 e)
	controlled by IETF RFC 3413	12.14.6.1.3 f)

*This object is an INDEX of the table in which it resides.

17.2.8 Structure of the IEEE8021-PBB MIB

The IEEE8021-PBB MIB provides objects to configure and manage provider backbone bridges.

Objects in this MIB module are arranged based on their description in Clause 12. Where appropriate in the table, the corresponding Clause 12 management reference is included.

Table 17-13 indicates the structure of the IEEE8021-PBB MIB module.

Table 17-13—IEEE8021-PBB MIB structure and relationship to this standard

Variable	Clause 17 MIB table/group	Reference
Backbone Edge Bridge Configuration	ieee8021PbbBackboneEdgeBridgeObjects	12.16.1.1, 12.16.1.2
	ieee8021PbbBackboneEdgeBridgeAddress	
	ieee8021PbbBackboneEdgeBridgeName	
	ieee8021PbbNumberOfIComponents	
	ieee8021PbbNumberOfBComponents	
	ieee8021PbbNumberOfBebPorts	
	ieee8021PbbNextAvailablePipIfIndex	
Virtual Instance Port Configuration	ieee8021PbbVipTable	12.16.2.1
	ieee8021BridgeBasePortComponentId*	
	ieee8021BridgeBasePort*	
	ieee8021PbbVipPipIfIndex	
	ieee8021PbbVipISid	
	ieee8021PbbVipDefaultDstMAC	
	ieee8021PbbVipType	
	ieee8021PbbVipRowStatus	

Table 17-13—IEEE8021-PBB MIB structure and relationship to this standard (continued)

Variable	Clause 17 MIB table/group	Reference
I-SID to VIP Cross Reference	ieee8021IsidToVipTable	12.16.3.1, 12.16.3.2
	ieee8021PbbISidToVipISid*	
	ieee8021PbbISidToVipComponentId	
	ieee8021PbbISidToVipPort	
Provider Instance Port Configuration	ieee8021PbbPipTable	12.16.4.1, 12.16.4.2
	ieee8021PbbPipIfIndex*	
	ieee8021PbbPipBMACAddress	
	ieee8021PbbPipName	
	ieee8021PbbPipIComponentId	
	ieee8021PbbPipVipMap	
	ieee8021PbbPipVipMap1	
	ieee8021PbbPipVipMap2	
	ieee8021PbbPipVipMap3	
	ieee8021PbbPipVipMap4	
	ieee8021PbbPipRowStatus	
Provider Instance Port Priority	ieee8021PbbPipPriorityTable	12.16.4.1, 12.16.4.2
	(AUGMENTS ieee8021PbbPipEntry)	
	ieee8021PbbPipPriorityCodePointSelection	12.16.4.5, 12.16.4.6
	ieee8021PbbPipUseDEI	12.16.4.11, 12.16.4.12
	ieee8021PbbPipRequireDropEncoding	12.16.4.13, 12.16.4.14
Provider Instance Port Traffic Class	ieee8021PbbPipTrafficClassTable	
	ieee8021PbbPipIfIndex*	
	ieee8021PbbPipTrafficClassPriority*	
	ieee8021PbbPipTrafficClass	
PIP Priority Code Point Decoding Table	ieee8021PbbPipDecodingTable	12.16.4.7, 12.16.4.8
	ieee8021PbbPipIfIndex*	

Table 17-13—IEEE8021-PBB MIB structure and relationship to this standard (continued)

Variable	Clause 17 MIB table/group	Reference
	ieee8021PbbPipDecodingPriorityCodePointRow*	
	ieee8021PbbPipDecodingPriorityCodePoint*	
	ieee8021PbbPipDecodingPriority	
	ieee8021PbbPipDecodingDropEligible	
PIP Priority Code Point Encoding Table	ieee8021PbbPipEncodingTable	12.16.4.9, 12.16.4.10
	ieee8021PbbPipIfIndex*	
	ieee8021PbbPipEncodingPriorityCodePointRow*	
	ieee8021PbbPipEncodingPriorityCodePoint*	
	ieee8021PbbPipEncodingDropEligible*	
	ieee8021PbbPipEncodingPriority	
VIP to PIP Cross Reference	ieee8021PbbVipToPipMappingTable	12.16.4.3, 12.16.4.4
	ieee8021BridgeBasePortComponentId*	
	ieee8021BridgeBasePort*	
	ieee8021PbbVipToPipMappingPipIfIndex	
	ieee8021PbbVipToPipMappingStorageType	
	ieee8021PbbVipToPipMappingRowStatus	
Customer Backbone Port Configuration	ieee8021PbbCBPSERVICEmappingTable	12.16.5.1, 12.16.5.2
	ieee8021BridgeBasePortComponentId*	
	ieee8021BridgeBasePort*	
	ieee8021PbbCBPSERVICEmappingBackboneSid*	
	ieee8021PbbCBPSERVICEmappingBVid	
	ieee8021PbbCBPSERVICEmappingDefaultBackboneDest	
	ieee8021PbbCBPSERVICEmappingType	
	ieee8021PbbCBPSERVICEmappingLocalSid	
	ieee8021PbbCBPSERVICEmappingRowStatus	
Customer Backbone Port creation/ deletion	ieee8021PbbCbpTable	17.5.3.4

Table 17-13—IEEE8021-PBB MIB structure and relationship to this standard (continued)

Variable	Clause 17 MIB table/group	Reference
	ieee8021BridgeBasePortComponentId*	
	ieee8021BridgeBasePort*	
	ieee8021PbbCbpRowStatus	

*This object is an INDEX of the table in which it resides.

17.2.9 Structure of the IEEE8021-DDCFM MIBs

The IEEE8021-DDCFM MIB provides objects to configure and manage the DDCFM capabilities.

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects as follows:

- a) Reflection Responder subtree: this subtree contains the objects that are applicable to bridges that support DDCFM's RR function.
- b) RFM Receiver subtree: this subtree contains the objects that are applicable to bridges or end stations that support DDCFM's RFM Receiver function.
- c) Decapsulator Responder subtree: this subtree contains the objects that are applicable to bridges that support DDCFM's DR function.
- d) SFM Originator subtree: this subtree contains the objects that are applicable to bridges or end stations that support DDCFM's SFM Originator function.

Table 17-14 indicates the structure of the IEEE8021-DDCFM MIB module.

Table 17-14—IEEE8021-DDCFM MIB structure and relationship to this standard

Variable	Clause 17 MIB Table/Group	Reference
	ieee8021DdcfmStackTable	DDCFM Stack managed object (12.17.1)
	ieee8021DdcfmStackIfIndex	12.17.1.1.2 a)
	ieee8021DdcfmStackRrMdLevel	12.17.1.1.3 b1)
	ieee8021DdcfmStackRrDirection	
	ieee8021DdcfmStackRFMreceiverMdLevel	12.17.1.1.3 b2)
	ieee8021DdcfmStackDrMdLevel	12.17.1.1.3 b3)
	ieee8021DdcfmStackDrVlanIdOrNone	
	ieee8021DdcfmStackSFMOiginatorMdLevel	12.17.1.1.3 b4)
	ieee8021DdcfmStackSFMOiginatorVlanIdOrNone	
	ieee8021DdcfmStackSFMOiginatorDirection	
	ieee8021DdcfmRrTable	Reflection Responder managed object (12.17.2)

Table 17-14—IEEE8021-DDCFM MIB structure and relationship to this standard (*continued*)

Variable	Clause 17 MIB Table/Group	Reference
	ieee8021DdcfmRr	
	ieee8021DdcfmRrIfIndex	12.17.2.1.2 a1)
	ieee8021DdcfmRrMdIndex	12.17.2.1.2 a2)
	ieee8021DdcfmRrDirection	12.17.2.1.2 a3)
	ieee8021DdcfmRrPrimaryVlanIdOrNone	12.17.2.2.2 b1)
	ieee8021DdcfmRrFilter	12.17.2.2.2 b2)
	ieee8021DdcfmRrSamplingInterval	12.17.2.2.2 b3)
	ieee8021DdcfmRrTargetAddress	12.17.2.2.2 b4)
	ieee8021DdcfmRrContinueFlag	12.17.2.2.2 b5)
	ieee8021DdcfmRrDuration	12.17.2.2.2 b7)
	ieee8021DdcfmRrDurationInTimeFlag	12.17.2.2.2 b6)
	ieee8021DdcfmRrVlanPriority ieee8021DdcfmRrVlanDropEligible	12.17.2.2.2 b9)
	ieee8021DdcfmRrFloodingFlag	12.17.2.2.2 b10)
	ieee8021DdcfmRrTruncationFlag	12.17.2.2.2 b.11
ReflectionRequest (29.3.1.15)	ieee8021DdcfmRrActivationStatus	12.18.1.2.3 b12)
RRwhile (29.3.1.9)	ieee8021DdcfmRrRemainDuration	12.17.2.3.3 b13)
nextRFMtransID (29.3.1.16)	ieee8021DdcfmRrNextRfmTransID	12.17.2.3.3 b14)
ieee8021DdcfmRFMReceiverTable		RFM Receiver managed object (12.17.3)
	ieee8021DdcfmRFMReceiver	
	ieee8021DdcfmRfmReceiverIfIndex ieee8021DdcfmRfmReceiverMdIndex	12.18.2.1.2 a2)
ieee8021DdcfmDrTable		Decapsulator Responder managed object (12.17.4)
	ieee8021DdcfmDr	
	ieee8021DdcfmDrIfIndex ieee8021DdcfmDrMdIndex ieee8021DdcfmDrVlanIdOrNone	12.17.3.1.2 a2)
	ieee8021DdcfmDrSourceAddressStayFlag	12.17.4.3.2 b1)
	ieee8021DdcfmDrSfmOriginator	12.17.4.3.2 b2)
	ieee8021DdcfmDrFloodingFlag	12.17.4.3.2 b3)
DRtime (29.3.8.2)	ieee8021DdcfmDrDuration	12.17.4.3.2 b4)

Table 17-14—IEEE8021-DDCFM MIB structure and relationship to this standard (*continued*)

Variable	Clause 17 MIB Table/Group	Reference
DRactive (29.3.8.4)	ieee8021DdcfDrActivationStatus	12.17.3.2.3 b6)
DRwhile (29.3.8.1)	ieee8021DdcfDrRemainDuration	12.17.3.2.3 b7)
SFMsequenceErrors (29.3.8.7)	ieee8021DdcfDrSFMsequenceErrors	12.17.4.2.3 b8)
	ieee8021DdcfSoTable	SFM Originator managed object (12.17.5)
	ieee8021DdcfSFMOriginator	
	ieee8021DdcfSfmIfIndex ieee8021DdcfSfmMdIndex ieee8021DdcfSfmSoVlanIdOrNone ieee8021DdcfSfmSoDirection	12.17.4.1.2 a2)
	ieee8021DdcfSoDrMacAddress	12.17.5.4.3 b2)
	ieee8021DdcfSoDuration	12.17.5.4.3 b3)
	ieee8021DdcfSoActivationStatus	12.17.4.2.3 b4)
	ieee8021DdcfSoRemainDuration	12.17.4.2.3 b5)

17.2.10 Structure of the IEEE8021-PBBTE MIB

The IEEE8021-PBBTE MIB provides objects to configure and manage Provider Backbone Bridges that support Traffic Engineering.

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. Where appropriate, the corresponding Clause 12 management reference is also included.

Table 17-15 shows the mapping of the PBB-TE Clause 12 managed objects to the tables and columns of the PBB-TE MIB that model those managed objects.

Table 17-15—IEEE8021-PBBTE MIB Structure and relationship to this standard

Variable	IEEE MIB table/object	Reference
	ieee8021PbbTeProtectionGroupListTable	
	ieee8021PbbTeProtectionGroupListGroupId	12.18.1
	ieee8021PbbTeProtectionGroupListWorkingMA	12.18.1.1.3 a)
	ieee8021PbbTeProtectionGroupListProtectionMA	12.18.1.1.3 b)
	ieee8021PbbTeMASharedGroupTable	
	ieee8021PbbTeMASharedGroupId	12.18.1.1.3 c)
	ieee8021PbbTeTesiTable	

Table 17-15—IEEE8021-PBBTE MIB Structure and relationship to this standard (continued)

Variable	IEEE MIB table/object	Reference
	ieee8021PbbTeTesiComponent	12.16.5.3.2 a)
	ieee8021PbbTeTesiBridgePort	12.16.5.3.2 b)
	ieee8021PbbTeTeSiEspTable	
	ieee8021PbbTeTeSiEspEspIndex	12.16.5.3.2 c)
	ieee8021PbbTeTeSiEspEsp	12.16.5.3.2 c)
	ieee8021PbbTeProtectionGroupConfigTable	
	ieee8021PbbTeProtectionGroupConfigState	12.18.2.1.3 d)
	ieee8021PbbTeProtectionGroupConfigCommandStatus	12.18.2.1.3 d)
	ieee8021PbbTeProtectionGroupConfigCommandLast	12.18.2.1.3 b)
	ieee8021PbbTeProtectionGroupConfigAdmin	12.18.2.3.2 b)
	ieee8021PbbTeProtectionGroupConfigActiveRequests	12.18.2.1.3 d)
WTRwhile (26.10.3.2.1)	ieee8021PbbTeProtectionGroupConfigWTR	12.18.2.1.3 e)
HoldOffWhile (26.10.3.2.2)	ieee8021PbbTeProtectionGroupConfigHoldOff	12.18.2.1.3 f)
	ieee8021PbbTeProtectionGroupISidTable	
	ieee8021PbbTeProtectionGroupISidGroupId	12.18.2.1.3 b)
	ieee8021PbbTeBridgeStaticForwardAnyUnicastTable	
	ieee8021PbbTeBridgeStaticForwardAnyUnicastVlanIndex	8.8.1
	ieee8021PbbTeBridgeStaticForwardAnyUnicastEgressPorts	8.8.1
	ieee8021PbbTeBridgeStaticForwardAnyUnicastForbiddedPorts	8.8.1

This standard defines a TE-SID as an implementation dependent identifier used to refer to a TESI. However, TE-SIDs are used externally by management to refer to these TESIs; in particular they are used to create maintenance associations (MAs) that monitor the health of a TESI. Thus, while the identifier itself is implementation dependent, a canonical external representation of the identifiers is needed for the purpose of MIB definition. The textual convention `IEEE8021PbbTeTSidId` is used to represent TE-SID values externally. This convention defines TE-SIDs to be a simple integer of local significance to a particular bridge component for the purpose of MIB management.

A TESI consists of a set of ESPs. Each ESP is identified by a 3 tuple consisting of an <ESP-DA,ESP-SA,ESP-VID>. These tuples are represented by the textual convention `IEEE8021PbbTeEsp`, which is a fixed length octet string containing the two MAC addresses along with the VID value.

TESIs themselves are represented by the `ieee8021PbbTeTeSidTable`. This table has five columns. Two are indices and consist of the TE-SID and an index into the list of ESPs. The remaining columns consist of the ESP itself along with a storage type and row status. The storage type indicates if the value defined by this row of the MIB persists across system restarts. The row status column is used by the management station to add, delete, activate, and deactivate rows from this table. Consider Figure 26-7 of this standard.

The `ieee8021PbbTeTeSidTable` for that switch would have the information shown in Table 17-16, assuming no other TESIs were configured. In the example shown in Table 17-16, two TESIs are present in the system. The first has a single ESP, while the second has two ESPs.

Table 17-16—Example of `ieee8021PbbTeTeSidTable`

Id	EspIndex	Esp	StorageType	RowStatus
1	1	<DA3, SA1, VID-1>	nonVolatile	Active
2	1	<DA3, SA2, VID-1>	nonVolatile	Active
2	2	<DA3, SA3, VID-2>	nonVolatile	Active

17.2.10.1 Using the MIB to create MAs associated with TESIs

In order to support protection switching, PBB-TE defines a mechanism to allow redundant TESIs to carry the same traffic. The implementation of protection switching requires a mechanism to monitor the health of this TESI. The CFM Maintenance association is used as this mechanism.

Configuring an MA requires adding a row to the CFM MIB's StackTable. Two columns are used in the StackTable to define the data stream upon which the MA operates. These are the `IEEE8021ServiceSelectorType` column and the `IEEE8021ServiceSelectorValue` column. The first column defines how the value in the second column is to be interpreted. If the value of the first column is set to `tesid(3)`, then the second column is interpreted as the external representation of a TE-SID.

Thus, TESIs, and their corresponding identifiers, configured in the `ieee8021PbbTeTeSidTable` provide the TE-SID values that are used as parameters to rowcreate operations for the CFMStackTable to create MAs that monitor the health of TESIs in a PBB TE network. Creating entries in the CFMStackTable, in turn, defined MAID values, represented as unsigned integers, that are used subsequently to configure the protected service.

17.2.10.2 Using the MIB to create protection groups

A TE protection group is a group of two TESIs referred to as WORKING and PROTECTION between a pair of CBPs. The TE protection group carries a set of backbone service instances between these CBPs and in the event of failure of one of the TESIs, the other TESI is used to transport the backbone service instances.

The TE protection groups are created by adding rows to the `ieee8021PbbTeProtectionGroupListTable`. Each entry in the table contains two columns indicating the MAs associated with the TESIs used to carry the traffic and a RowStatus column used to managed the creation and deletion of the TE protection group. The MAs are referred to by MAID and can be found in the StackTable of the CFM MIB managed objects.

17.2.10.3 Using the MIB to query and configure protection groups

Each configuration group has status and configuration information that can be managed. This information is located in the `ieee8021PbbTeProtectionGroupConfigTable`. This table allows management stations to determine which TESI is being used to transmit traffic and it also allows for management commands to switch the traffic between the two TESIs. There are also optional parameters that configure the time to trigger a switchover and the time to restore after a failure condition has cleared.

17.2.10.4 Using the MIB to prevent erroneous forwarding in the PBB-TE network

Normally in a bridged network, a frame with an unknown unicast destination address is flooded to an appropriate set of ports determined by the frame's VID. PBB-TE networks do not operate this way. A frame with an unknown unicast destination MAC address whose VID is an ESP-VID is discarded. Item a) 3) of 8.8.1 takes care of this. This clause allows for a new kind of filtering entry. This filtering entry applies to all frames with individual, as opposed to group or broadcast, destination addresses to which no other filtering entry applies. By associating a drop action with this filtering action, the relay function will cause all frames that arrive with a VID belonging to the set of PBB-TE VIDs with an unknown DA to be dropped. The table that sets up these filtering entries is called the `ieee8021PbbTeBridgeStaticForwardAnyUnicastTable`.

17.2.11 Structure of the TPMR MIB

The TPMR MIB is organized into the following MIB groups:

- TPMR Configuration
- TPMR Port Configuration
- TPMR Port Counters
- TPMR Port Discard Details
- TPMR MSP Configuration
- TPMR MSP Statistics

Clause 12 defines the information model associated with this standard in a protocol-independent manner. Table 17-17 describes the relationship between the SMIv2 objects defined in this MIB module and the protocol-independent objects defined in Clause 12.

Table 17-17—IEEE8021-TPMR MIB Structure and relationship to this standard

Managed object definition	MIB table/object	Reference
TPMR name TPMR uptime	system (SNMPv2-MIB) sysName sysUpTime	12.19.1.1.1.3:a and 12.19.1.1.2.2:a 12.19.1.1.1.3:c
TPMR port name TPMR port type TPMR port MAC address	ifTable (IF-MIB) ifName ifType ifPhysAddress	12.19.1.2.1.3:a and 12.19.1.2.2.2:b 12.19.1.2.1.3:b 12.19.1.1.1.3:b,2
TPMR port number TPMR port management MAC address TPMR port management MAC address forwarding	ieee8021TpmpPortTable ieee8021TpmpPortNumber ieee8021TpmpPortMgmtAddr ieee8021TpmpPortMgmtAddrForwarding	12.19.1.1.1.3:b,1 12.19.1.1.1.3:b,3 12.19.1.2.1.3:c

Table 17-17—IEEE8021-TPMR MIB Structure and relationship to this standard (continued)

Managed object definition	MIB table/object	Reference
TPMR port Rx Frames TPMR port Rx Octets TPMR port Forwarded Frames TPMR port Discarded Frames TPMR port Discarded Frames Queue Full TPMR port Discarded Frames Lifetime TPMR port Discarded Frames Error	ieee8021TpmpPortStatsTable AUGMENTS ieee8021TpmpPortTable ieee8021TpmpPortStatsRxFrames ieee8021TpmpPortStatsRxOctets ieee8021TpmpPortStatsFramesForwarded ieee8021TpmpPortStatsFramesDiscarded ieee8021TpmpPortStatsFramesDiscardedQueueFull ieee8021TpmpPortStatsFramesDiscardedLifetime ieee8021TpmpPortStatsFramesDiscardedError	12.19.3.1.1.3:a 12.19.3.1.1.3:b 12.19.3.1.1.3:d 12.19.3.1.1.3:c 12.19.3.1.1.3:e 12.19.3.1.1.3:f 12.19.3.1.1.3:g
TPMR port discard details: source address TPMR port discard details: error reason	ieee8021TpmpPortDiscardDetailsTable ieee8021BridgeBasePortComponentId ^a ieee8021TpmpPortNumber [*] ieee8021TpmpPortDiscardDetailsIndex [*] ieee8021TpmpPortDiscardDetailsSource ieee8021TpmpPortDiscardDetailsReason	12.19.3.1.1.3:h 12.19.3.1.1.3:h
TPMR MSP link notify TPMR MSP link notify wait TPMR MSP link notify retry TPMR MSP MAC notify TPMR MSP MAC notify time TPMR MSP MAC notify recover	ieee8021TpmpMspTable AUGMENTS ieee8021TpmpPortTable ieee8021TpmpMspLinkNotify ieee8021TpmpMspLinkNotifyWait ieee8021TpmpMspLinkNotifyRetry ieee8021TpmpMspMacNotify ieee8021TpmpMspMacNotifyTime ieee8021TpmpMspMacRecoverTime	12.19.4.1.1.3:a and 12.19.4.1.2.2:b 12.19.4.1.1.3:b and 12.19.4.1.2.2:c 12.19.4.1.1.3:c and 12.19.4.1.2.2:d 12.19.4.1.1.3:d and 12.19.4.1.2.2:e 12.19.4.1.1.3:e and 12.19.4.1.2.2:f 12.19.4.1.1.3:f and 12.19.4.1.2.2:g
TPMR MSP tx acks TPMR MSP tx add notifications TPMR MSP tx add conformations TPMR MSP tx loss notifications TPMR MSP tx loss conformations TPMR MSP rx acks TPMR MSP rx add notifications TPMR MSP rx add conformations TPMR MSP rx loss notifications TPMR MSP rx loss conformations TPMR MSP add events TPMR MSP loss events TPMR MSP MAC status notifications	ieee8021TpmpMspStatsTable AUGMENTS ieee8021TpmpPortTable ieee8021TpmpMspStatsTxAcks ieee8021TpmpMspStatsTxAddNotifications ieee8021TpmpMspStatsTxAddConfirmations ieee8021TpmpMspStatsTxLossNotifications ieee8021TpmpMspStatsTxLossConfirmations ieee8021TpmpMspStatsRxAcks ieee8021TpmpMspStatsRxAddNotifications ieee8021TpmpMspStatsRxAddConfirmations ieee8021TpmpMspStatsRxLossNotifications ieee8021TpmpMspStatsRxLossConfirmations ieee8021TpmpMspStatsAddEvents ieee8021TpmpMspStatsLossEvents ieee8021TpmpMspStatsMacStatusNotifications	12.19.4.1.3.3:a 12.19.4.1.3.3:b 12.19.4.1.3.3:c 12.19.4.1.3.3:d 12.19.4.1.3.3:e 12.19.4.1.3.3:f 12.19.4.1.3.3:g 12.19.4.1.3.3:h 12.19.4.1.3.3:i 12.19.4.1.3.3:j 12.19.4.1.3.3:k 12.19.4.1.3.3:l 12.19.4.1.3.3:m

^aThis object is an INDEX of the table in which it resides.

17.2.12 Structure of the IEEE8021-FQTSS MIB

The IEEE8021-FQTSS MIB provides objects to configure and manage those aspects of a VLAN Bridge that are related to forwarding and queuing for time-sensitive streams (see Clause 34).

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. Where appropriate, the corresponding Clause 12 management reference is also included.

Table 17-18 indicates the structure of the IEEE8021-FQTSS MIB module.

Table 17-18—FQTSS MIB structure and object cross reference

MIB table	MIB object	References
<i>ieee8021FqtssBap subtree</i>		
ieee8021FqtssBapTable		Bandwidth Availability Parameter Table, 12.20.1, 34.3
	ieee8021FqtssBAPTrafficClass	Traffic class (Table index)
	ieee8021FqtssDeltaBandwidth	deltaBandwidth, 12.20.1, 34.3
	ieee8021FqtssOperIdleSlopeMs	operIdleSlope, 12.20.1, 34.3
	ieee8021FqtssOperIdleSlopeLs	operIdleSlope, 12.20.1, 34.3
	ieee8021FqtssAdminIdleSlopeMs	adminIdleSlope, 12.20.1, 34.3
	ieee8021FqtssAdminIdleSlopeLs	adminIdleSlope, 12.20.1, 34.3
<i>ieee8021FqtssMappings subtree</i>		
ieee8021FqtssTxSelectionAlgorithmTable		Transmission Selection Algorithm Table, 12.20.2, 8.6.8
	ieee8021FqtssTrafficClass	Traffic class (Table index)
	ieee8021FqtssTxSelectionAlgorithmID	Transmission selection algorithm, 12.20.2, 8.6.8
ieee8021FqtssSrpRegenOverrideTable		SRP domain boundary port priority regeneration override table, 12.20.3, 6.6.4, 6.9.4
	ieee8021FqtssSrClassPriority	Received priority (Table index)
	ieee8021FqtssPriorityRegenOverride	Regenerated priority, 12.20.3, 6.9.4
	ieee8021FqtssSrpBoundaryPort	SRPdomainBoundaryPort, 12.20.3, 6.6.4

17.2.13 Structure of the Congestion Notification MIB

Subclause 12.21 of this document defines the information model associated with this standard in a protocol-independent manner. Table 17-19 describes the relationship between the SMIv2 objects defined in the MIB module in 17.7.13 and the variables and managed objects defined in Clause 12 and Clause 32.

Table 17-19—Variables, managed object tables, and MIB objects

Clause 32 table or variable	Clause 12/ Clause 32 reference	MIB object (17.7)
CN component managed object, CN component variables	12.21.1 32.2	ieee8021CnGlobalTable
cngMasterEnable	32.2.1	ieee8021CnGlobalMasterEnable
cngCnmTransmitPriority	32.2.2	ieee8021CnGlobalCnmTransmitPriority
cngDiscardedFrames	32.2.3	ieee8021CnGlobalDiscardedFrames
cngErroredPortList	32.2.4	ieee8021CnErroredPortTable
CN component priority managed object, Congestion notification per-CNPV variables	12.21.2 32.3	ieee8021CnComptnPriTable
cncpDefModeChoice	32.3.1	ieee8021CnComPriDefModeChoice
cncpAlternatePriority	32.3.2	ieee8021CnComPriAlternatePriority
cncpAutoAltPri	32.3.3	ieee8021CnComPriAutoAltPri
cncpAdminDefenseMode	32.3.4	ieee8021CnComPriAdminDefenseMode
cncpCreation	32.3.5	ieee8021CnComPriCreation
cncpLldpInstanceChoice	32.3.6	ieee8021CnComPriLldpInstanceChoice
cncpLldpInstanceSelector	32.3.7	ieee8021CnComPriLldpInstanceSelector
CN Port priority managed object, CND defense per-Port per-CNPV variables	12.21.2 32.4	ieee8021CnPortPriTable
cnpdDefModeChoice	32.4.1	ieee8021CnPortPriDefModeChoice
cnpdAdminDefenseMode	32.4.2	ieee8021CnPortPriAdminDefenseMode
cnpdAutoDefenseMode	32.4.3	ieee8021CnPortPriAutoDefenseMode
cnpdLldpInstanceChoice	32.4.4	ieee8021CnPortPriLldpInstanceChoice
cnpdLldpInstanceSelector	32.4.5	ieee8021CnPortPriLldpInstanceSelector
cnpdAlternatePriority	32.4.6	ieee8021CnPortPriAlternatePriority

Table 17-19—Variables, managed object tables, and MIB objects (continued)

Clause 32 table or variable	Clause 12/ Clause 32 reference	MIB object (17.7)
Congestion Point managed object, Congestion Point variables	12.21.4 32.8	ieee8021CnCpTable
cpMacAddress	32.8.1	ieee8021CnCpMacAddress
cpld	32.8.2	ieee8021CnCpIdentifier
cpQSp	32.8.3	ieee8021CnCpQueueSizeSetPoint
cpW	32.8.6	ieee8021CnCpFeedbackWeight
cpSampleBase	32.8.11	ieee8021CnCpMinSampleBase
cpDiscardedFrames	32.8.12	ieee8021CnCpDiscardedFrames
cpTransmittedFrames	32.8.13	ieee8021CnCpTransmittedFrames
cpTransmittedCnms	32.8.14	ieee8021CnCpTransmittedCnms
cpMinHeaderOctets	32.8.15	ieee8021CnCpMinHeaderOctets
(None) ^a	(none)	ieee8021CnCpidToInterfaceTable
Reaction Point port priority managed object, Reaction Point per-Port per-CNPV variables	12.21.5 32.10	ieee8021CnRpPortPriTable
rpppMaxRps	32.10.1	ieee8021CnRpPortPriMaxRps
rpppCreatedRps	32.10.2	ieee8021CnRpPortPriCreatedRps
rpppRpCentiseconds	32.10.3	ieee8021CnRpPortPriCentiseconds
Reaction Point group managed object, Reaction Point group variables	12.21.6 32.11	ieee8021CnRpGroupTable
rpgEnable	32.11.1	ieee8021CnRpgEnable
rpgTimeReset	32.11.2	ieee8021CnRpgTimeReset
rpgByteReset	32.11.3	ieee8021CnRpgByteReset
rpgThreshold	32.11.4	ieee8021CnRpgThreshold
rpgMaxRate	32.11.5	ieee8021CnRpgMaxRate
rpgAiRate	32.11.6	ieee8021CnRpgAiRate
rpgHaiRate	32.11.7	ieee8021CnRpgHaiRate
rpgGd	32.11.8	ieee8021CnRpgGd
rpgMinDecFac	32.11.9	ieee8021CnRpgMinDecFac
rpgMinRate	32.11.10	ieee8021CnRpgMinRate

^aThis table is an artifact of SNMP, required to find an entry in the ieee8021CnCpTable, given a CPID (32.8.2, 33.4.4).

17.2.14 Structure of the IEEE8021-SRP MIB

The IEEE8021-SRP MIB provides objects to configure and manage those aspects of a VLAN Bridge that are related to the Stream Reservation Protocol (SRP).

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. Where appropriate, the corresponding Clause 12 management reference is also included.

Table 17-20 indicates the structure of the IEEE8021-SRP MIB module.

Table 17-20—SRP MIB structure and object cross reference

MIB table	MIB object	References
<i>ieee8021SrpConfiguration subtree</i>		
	ieee8021SrpBridgeBaseTable	SRP control and status information for a bridge. Augments ieee8021BridgeBaseEntry.
	ieee8021SrpBridgeBaseMsrpEnabledStatus	Is MSRP enabled on this device? True or False, 12.22.1, 35.2.1.4d
	ieee8021SrpBridgeBaseMsrpTalkerPruning	talkerPruning, 12.22.1, 35.2.1.4b
	ieee8021SrpBridgeBaseMsrpMaxFanInPorts	msrpMaxFanInPorts, 12.22.1, 35.2.1.4(f)
	ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize	msrpLatencyMaxFrameSize, 12.22.1, 35.2.1.4(g)
	ieee8021SrpBridgePortTable	SRP Control and Status information for each port on the Bridge. Augments ieee8021BridgeBasePortEntry.
	ieee8021SrpBridgePortMsrpEnabledStatus	Is MSRP enabled on this port? True or False, 12.22.2, 35.2.1.4e.
	ieee8021SrpBridgePortMsrpFailedRegistrations	How many failed registrations have there been on this port, 10.7.12.1, 12.22.2.
	ieee8021SrpBridgePortMsrpLastPduOrigin	Source MAC Address of last MSRPDU received on this port, 10.7.12.2, 12.22.2.
	ieee8021SrpBridgePortSrPvid	Default VLAN ID for Streams on this port, Table 9-2, Table 12-11, 12.22.2, 35.2.1.4(i)
<i>ieee8021SrpLatency subtree</i>		
	ieee8021SrpLatencyTable	Maximum port latency per traffic class, 12.22.3, 35.2.2.8.6
	ieee8021SrpTrafficClass	Traffic class (Table index)
	ieee8021SrpPortTcLatency	Maximum port latency for the associated traffic class, 12.22.3, 35.2.1.4a, 35.2.2.8.6
<i>ieee8021SrpStreams subtree</i>		

Table 17-20—SRP MIB structure and object cross reference (continued)

MIB table	MIB object	References
ieee8021SrpStreamTable		Components that define the characteristics of a Stream., 12.22.4, 35.2.2.8.
	ieee8021SrpStreamID	StreamID (Table index), 12.22.4, 35.2.2.8.2.
	ieee8021SrpStreamDestinationAddress	Stream destination MAC address, 12.22.4, 35.2.2.8.3a.
	ieee8021SrpStreamVlanID	VLAN ID for Stream (0=default), 12.22.4, 35.2.2.8.3b.
	ieee8021SrpStreamTspecMaxFrameSize	Maximum frame size sent by Talker, 12.22.4, 35.2.2.8.4a.
	ieee8021SrpStreamTspecMaxIntervalFrames	Maximum number of frames sent per class measurement interval, 12.22.4, 35.2.2.8.4b, 34.4.
	ieee8021SrpStreamDataFramePriority	The Priority Code Point (PCP) value that the data Stream will be tagged with, 12.22.4, 35.2.2.8.5a.
	ieee8021SrpStreamRank	Emergency/nonemergency Rank associated with the Stream, 12.22.4, 35.2.2.8.5b.
<i>ieee8021SrpReservations subtree</i>		
ieee8021SrpReservationsTable		A table containing Stream attribute registrations per port, 12.22.5, 35.2.4.
	ieee8021SrpReservationStreamId	StreamID (Table index), 12.22.5, 35.2.2.8.2.
	ieee8021SrpReservationDirection	Talker or Listener (Table index), 12.22.5, 35.2.1.2.
	ieee8021SrpReservationDeclarationType	Advertise or Failed for Talkers. Asking Failed, Ready or Ready Failed for Listeners. 12.22.5, 35.2.1.3.
	ieee8021SrpReservationAccumulatedLatency	Latency at ingress port for Talker registrations, or latency at end of egress media for Listener Declarations, 12.22.5, 35.2.2.8.6.
	ieee8021SrpReservationFailureBridgeId	Bridge ID of Bridge that change Talker Advertise to Talker Failed, 12.22.5, 35.2.2.8.7a.
	ieee8021SrpReservationFailureCode	Failure Code associated with Bridge that changed Talker Advertise to Talker Failed, 12.22.5, 35.2.2.8.7b.
	ieee8021SrpReservationDroppedStreamFrames	Number of stream data frames (not MSRPDUs) that have been dropped for this stream on this port, 12.22.5, 35.2.5.1.
	ieee8021SrpReservationStreamAge	Number of seconds since reservation was established, 12.22.5, 35.2.1.4c.

17.3 Relationship to other MIBs

In order to facilitate interoperable management of VLAN Bridges by remote means, 17.7 contains a complete set of SMIv2 MIB modules.

Some of these MIB modules have been derived from, and now replace, previous IETF MIB modules, as noted in Table 17-1. This was necessary because significant changes had to be made in these MIB modules, in particular the table indexing schemes, to support the evolution of VLAN Bridges. For example, Provider Backbone Bridging, described in Clause 26, supports multiple Bridge instances within a Backbone Edge Bridge. To avoid duplication of the base Bridge MIB modules within a PBB MIB module, the base Bridge modules have been reindexed to allow their use within a Backbone Edge Bridge (BEB). A BEB may have multiple I-components and zero or one B-components. A component identifier index, unique within a particular BEB, is used in the various MIB tables of these modules to distinguish between instances. In Bridges supporting multiple component or Bridge instances, the component identifier index is still present but is set to a default value of 1.

17.3.1 Relationship of the IEEE8021-TC MIB to other MIB modules

Textual conventions (TCs) originally appeared in each of the MIB modules for IEEE Std 802.1Q produced by the IETF. However, with the transition of all modules for IEEE Std 802.1Q to this clause, it made sense to have a single module with all the TCs contained within it. Note that many of the original IETF TCs are still used in the various MIB modules for VLAN-Bridge management since they can be used unchanged.

These TCs in the IEEE8021-TC MIB module, as well as many TCs from IETF MIB modules, are used by all of the MIB modules presented in this clause.

The component identifier (IEEE8021PbbComponentIdentifierTC) is defined in this module and is used as the syntax for component ID table indices in subsequent MIB modules.

17.3.2 Relationship of the IEEE8021-BRIDGE MIB to other MIB modules

As described in Table 17-4 of 17.2.2, some IEEE 802.1D management objects have not been included in this MIB module because they overlap with objects in other MIB modules that are applicable to a bridge implementing this MIB module.

17.3.2.1 Relationship to the SNMPv2-MIB

The SNMPv2-MIB [RFC3418] defines objects that are generally applicable to managed devices. These objects apply to the device as a whole, irrespective of whether the device's sole functionality is bridging, or whether bridging is only a subset of the device's functionality.

As explained in Table 17-4 of 17.2.2, full support for the IEEE 802.1D management objects requires that the SNMPv2-MIB objects sysDescr and sysUpTime be implemented. Note that compliance with the current SNMPv2-MIB module requires additional objects and notifications to be implemented, as specified in IETF RFC 3418.

17.3.2.2 Relationship to the IF-MIB

The IF-MIB [RFC2863], requires that any MIB that is an adjunct of the IF-MIB clarify specific areas within the IF-MIB. These areas were intentionally left vague in the IF-MIB in order to avoid over-constraining the MIB, thereby precluding management of certain media types.

The IF-MIB enumerates several areas that a media-specific MIB must clarify. Each of these areas is addressed in a following subclause. The implementor is referred to the IF-MIB in order to understand the general intent of these areas.

The IF-MIB [RFC2863] defines managed objects for managing network interfaces. A network interface is thought of as being attached to a *subnetwork*. Note that this term is not to be confused with *subnet*, which refers to an addressing partitioning scheme used in the Internet suite of protocols. The term *segment* is used in this clause to refer to such a subnetwork, whether it be an Ethernet LAN, a *ring*, a WAN link, or even an SDH virtual circuit.

As explained in 17.2.2, full support for the IEEE 802.1D management objects requires that the IF-MIB objects ifIndex, ifType, ifDescr, ifPhysAddress, and ifLastChange are implemented. Note that compliance to the current IF-MIB module requires additional objects and notifications to be implemented as specified in IETF RFC 2863.

Implicit in this BRIDGE-MIB is the notion of Bridge Ports (3.22). Each Bridge Port is associated with one interface of the *interfaces* subtree (one row in ifTable), and in most situations, each Bridge Port is associated with a different interface. However, there are situations in which multiple Bridge Ports are associated with the same interface. That is, for a given IF-MIB, interface refers to one of the interface points in the bridging architecture (in Figure 8-1), and that zero or more multiple interface table entries can thus be instantiated for a given Bridge Port. An example of such a situation would be several Bridge Ports each corresponding one-to-one with several Ethernet private lines (or SDH virtual circuits) but all on the same interface. Alternatively, there is the link aggregation (IEEE Std 802.1AX) case where there are many physical Ports for one Bridge Port.

Each Bridge Port is uniquely identified by a Port Number. A Port Number has no mandatory relationship to an interface number, but in the simple case, a Port Number will have the same value as the corresponding interface's interface number. As a result of limitations in the BPDU for STP, this standard [see 12.3 i)] limits the maximum number of Bridge Ports to 4095. However, in the absence of spanning tree there is no restriction. As a result, Port Numbers are in the range (1..65535) to allow correspondence to interface numbers—but the STP restriction must be adhered to if it is supported on a bridge.

Discontinuities in the value of the counter on the Bridge Port can occur at reinitialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any).

Some entities perform other functionalities as well as bridging through the sending and receiving of data on their interfaces. In such situations, only a subset of the data sent/received on an interface is within the domain of the entity's bridging functionality. This subset is considered to be delineated according to a set of protocols, with some protocols being bridged, and other protocols not being bridged. For example, in an entity that exclusively performs bridging, all protocols would be considered as bridged, whereas in an entity that performs IP routing on IP datagrams and only bridges other protocols, only the non-IP data would be considered as having been bridged.

Thus, this BRIDGE-MIB (and in particular, its counters) are applicable only to that subset of the data on an entity's interfaces that is sent/received for a protocol being bridged. All such data is sent/received via the Ports of the bridge.

The BRIDGE MIB also defines a new interface type called an internal LAN (I-LAN). This interface is identified by an ifIndex and is intended to help manage internal associations between bridging components. The BRIDGE-MIB provides the ability to create and destroy I-LAN interfaces. When an I-LAN interface is created, a corresponding row is created in the ifTable. The ifType of this interface is 247.²⁷

²⁷IANA ifType registry is <http://www.iana.org/assignments/ianaiftype-mib>.

I-LAN interfaces support stacking through the ifStackTable. Through this mechanism, the I-LAN interface can create internal connections between components. For example, an I-LAN interface can connect two C-VLAN components in a system. To complete this process, the following steps are required:

- 1) Create C-VLAN component 1
- 2) Create C-VLAN Port 1 on component 1
- 3) Create a Bridge Port interface (ifType 209) identified by ifIndex 1 using a external mechanism
- 4) Associate C-VLAN Port 1 on component 1 to Bridge Port interface 1
- 5) Create an I-LAN interface identified with ifIndex 2
- 6) Stack Bridge Port interface 1 on the I-LAN interface 2
- 7) Create C-VLAN component 2
- 8) Create C-VLAN Port 1 on component 2
- 9) Create a Bridge Port interface (ifType 209) identified by ifIndex 3 using a external mechanism
- 10) Associate C-VLAN Port 1 on component 2 to Bridge Port interface 3
- 11) Stack Bridge Port interface 3 on the I-LAN interface 2

The result of the previous configuration steps are displayed in the Figure 17-1.

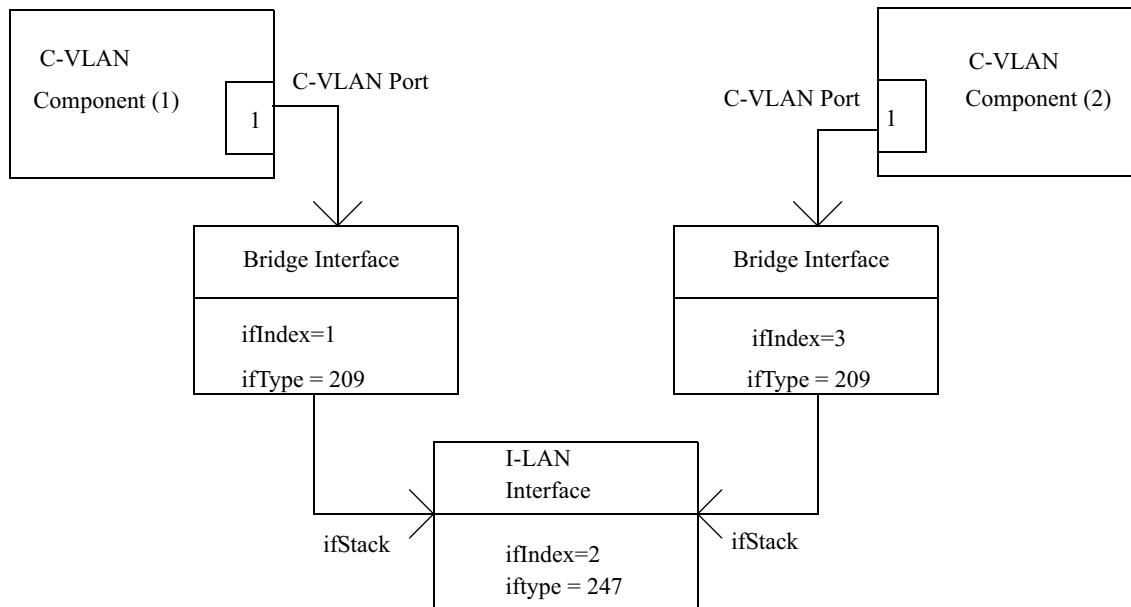


Figure 17-1—C-VLAN component internal LAN managed system

As a result, traffic can now be forwarded between C-VLAN component 1 and C-VLAN component 2 through the internal LAN.

17.3.2.3 FDB for IEEE 802.1D Bridges

The tables needed to explicitly manage the filtering database (FDB) for IEEE 802.1D Bridges have been removed from the IEEE8021-BRIDGE MIB. There were separate tables for this function in the original IETF MIBs, but these duplicate tables in the IEEE8021-QBRIDGE MIB and are not needed.

As a result, single filtering database implementations, like IEEE Std 802.1D, should just implement the ieee8021QBridgeTpFdbTable with an ieee8021QBridgeFdbId of 1, and setting any VID indices to 0.

17.3.3 Relationship of the IEEE8021-RSTP MIB to other MIB modules

The objects in the RSTP MIB supplement those defined in the BRIDGE MIB.

The original IETF BRIDGE-MIB [RFC1493] and its SMIv2-compliant version [RFC4188] have been replaced by 17.7.2. Besides using different prefix names for modules, an additional index was added to allow use of the RSTP MIB module in PBB.

17.3.4 Relationship of the IEEE8021-Q-BRIDGE MIB to other MIB modules

17.3.4.1 Relationship to the IF-MIB

This standard assumes the interpretation of the Interfaces Subtree to be in accordance with the IF-MIB [RFC2863], which states that the interfaces table (ifTable) contains information on the managed resource's interfaces and that each sublayer below the internetwork layer of a network interface is considered an interface.

This standard does not make any assumption that within an entity, VIDs that are instantiated as an entry in dot1qVlanCurrentTable—either by management configuration through dot1qVlanStaticTable or by dynamic means (e.g., through MVRP)—are also represented by an entry in ifTable.

Where an entity contains higher-layer protocol entities (e.g., IP-layer interfaces that transmit and receive traffic to/from a VLAN), these should be represented by an interface that represents the protocol entity and an interface of ifType l2vlan (135) with the ifStackTable indicating the stacking relationship between the two entities.

17.3.4.1.1 ifStackTable

In addition, the IF-MIB [RFC2863] defines a table 'ifStackTable' for describing the relationship between logical interfaces within an entity. It is anticipated that implementors will use this table to describe the binding of (for example) LAG interfaces to physical Ports, although the presence of VLANs makes the representation less than perfect for showing connectivity. The ifStackTable cannot represent the full capability of this standard, since this standard makes a distinction between VLAN bindings on *ingress* to and *egress* from a Port. These relationships may or may not be symmetrical; whereas, Interface MIB Evolution assumes a symmetrical binding for transmit and receive. This makes it necessary to define other manageable objects for configuring which Ports are in the member set for which VIDs.

17.3.4.1.2 ifRcvAddressTable

This table contains all MAC addresses, unicast, multicast, and broadcast, for which an interface will receive packets and forward them up to a higher-layer entity for local consumption. Note that this does not include addresses for data-link layer control protocols such as STP, RSTP, MSTP, MMRP, or MVRP. The format of the address, contained in ifRcvAddressAddress, is the same as for ifPhysAddress.

This table does not include unicast or multicast addresses that are accepted for possible forwarding out some other Port. This table is explicitly not intended to provide a Bridge address filtering mechanism.

17.3.4.2 Relationship to IEEE8021-BRIDGE MIB

This subclause defines how objects in the IEEE8021-BRIDGE MIB module should be represented for devices that implement the extensions. Some of the old objects are less useful in such devices, but must still be implemented for reasons of backwards compatibility.

17.3.4.2.1 ieee8021BridgeBase subtree

This subtree contains objects that are applicable to all types of Bridges. Interpretation of this subtree is unchanged.

17.3.4.2.2 ieee8021BridgeTp subtree

This subtree contains objects that describe the entity's state with respect to transparent bridging.

In a device operating with a single Filtering Database, interpretation of this subtree is unchanged.

In a device supporting multiple Filtering Databases, this subtree is interpreted as follows:

- a) ieee8021BridgeTpLearnedEntryDiscards
The number of times that any of the FDBs became full.
- b) ieee8021BridgeTpAgingTime
This applies to all Filtering Databases.
- c) ieee8021BridgeTpFdbTable
Report MAC addresses learned on each Port, regardless of which Filtering Database they have been learned in. If an address has been learned in multiple databases on a single Port, report it only once. If an address has been learned in multiple databases on more than one Port, report the entry on any one of the valid Ports.
- d) ieee8021BridgeTpPortTable
This table is Port-based and is not affected by multiple Filtering Databases or multiple VIDs. The counters should include frames received or transmitted for all VIDs.

17.3.4.2.3 ieee8021BridgeStatic subtree

This optional subtree contains objects that describe the configuration of destination-address filtering.

In a device operating with a single Filtering Database, interpretation of this subtree is unchanged.

In a device supporting multiple Filtering Databases, this subtree is interpreted as follows:

- a) ieee8021BridgeStaticTable
Entries read from this table include all static entries from all of the Filtering Databases. Entries for the same MAC address and receive Port in more than one Filtering Database must appear only once, since these are the indices of this table. This table should be implemented as read-only in devices that support multiple filtering databases. Instead, write access should be provided through dot1qStaticUnicastTable and dot1qStaticMulticastTable, as defined in this standard.

17.3.4.2.4 Additions to the IEEE8021-BRIDGE-MIB

To supplement the BRIDGE-MIB, this module contains the following:

- 1) Support for multiple traffic classes and dynamic multicast filtering as per IEEE Std 802.1D.
- 2) Support for bridged VLANs as per IEEE Std 802.1Q.
- 3) Support for 64-bit versions of BRIDGE-MIB Port counters.

17.3.5 Relationship of the IEEE8021-PB-BRIDGE MIB to other MIB modules

To supplement the Q-BRIDGE-MIB, this module contains the following:

- 1) Support for VID translation.

- 2) Support for Provider Edge Bridges.

17.3.6 Relationship of the IEEE8021-MSTP MIB to other MIB modules

The objects in the IEEE8021-MSTP MIB supplement those defined in the IEEE8021-RSTP MIB.

17.3.7 Relationship of the IEEE8021-CFM MIB to other MIB modules

17.3.7.1 Relationship to Interface MIB

Subclause 17.7.7 defines a CFM MIB module that supports 12.14 with the textual conventions imported from the TC MIB in 17.7.1. A system implementing the CFM MIB module in 17.7.7 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 other clauses that define the Bridge MIB modules. Even if a Bridge supports none of these, if it supports the CFM MIB module, and hence, the Interfaces Group, the clarifications from the other clauses shall be applied to the Interfaces Group. The relationship between IETF RFC 2863 and IETF RFC 3418 interfaces and Bridge Ports is also described in previous subclauses of 17.3. In addition, if both the CFM MIB module (17.7.7) and IEEE Std 802.1AX-2008 (IEEE Std 802.3, Clause 43), are supported, a Bridge shall

- a) Assign each IEEE 802.1AX 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 802.1AX Ports being aggregated.

17.3.7.2 IEEE8021-CFM and IEEE8021-CFM-V2 MIBs

The inclusion of Provider Backbone Bridge (PBB) (Clause 26) required the addition of additional indices to numerous tables in the CFM MIB. As a result, these tables are deprecated in the existing IEEE8021-CFM MIB module and new reindexed tables are created in a second IEEE8021-CFM-V2 MIB module. Unchanged tables remain in the IEEE8021-CFM MIB module. Specifically, groups with tables left unchanged are imported in the IEEE8021-CFM-V2 MIB module to appear in the compliance statement and new groups containing the new reindexed tables are added to the compliance statement.

The following tables are renamed:

```
dot1agPbbCfmStackTable
dot1agPbbCfmVlanTable
dot1agPbbCfmDefaultMdTable
dot1agPbbCfmConfigErrorListTable
dot1agPbbCfmMaCompTable
```

The new names are as follows:

```
ieee8021CfmStackTable
ieee8021CfmVlanTable
```

ieee8021CfmDefaultMdTable
ieee8021CfmConfigErrorListTable
ieee8021CfmMaCompTable

17.3.8 Relationship of the IEEE8021-PBB MIB to other MIB modules

The IF-MIB, [RFC2863], requires that any MIB that is an adjunct of the IF-MIB clarify specific areas within the IF-MIB. These areas were intentionally left vague in the IF-MIB in order to avoid over-constraining the MIB, thereby precluding management of certain media types. The IF-MIB enumerates several areas that a media-specific MIB must clarify. Each of these areas is addressed in a following subclause. The implementor is referred to the IF-MIB in order to understand the general intent of these areas. The IF-MIB [RFC2863] defines managed objects for managing network interfaces.

A network interface is considered attached to a *subnetwork*. The term *segment* is used to refer to such a subnetwork, whether it be an Ethernet LAN, a *ring*, a WAN link, or even an SDH virtual circuit. Full support for PBB managed objects requires that the IF-MIB objects ifIndex, ifType, ifDescr, ifPhysAddress, and ifLastChange be implemented. Note that compliance to the current IF-MIB module requires additional objects and notifications to be implemented as specified in IETF RFC 2863.

The interpretation of the Interfaces Subtree is assumed to be in accordance with the IF-MIB, which states that the interfaces table (ifTable) contains information on the managed resource's interfaces and that each sublayer below the internetwork layer of a network interface is considered an interface. Specifically, in the PBB MIB module, the PIP index has been equated to an interface index (ifIndex). Therefore, when a PIP is created, a row is also created in the ifTable identified by the PIP's ifIndex. The ifType of this interface is 248.²⁸

In addition, the IF-MIB defines a table 'ifStackTable' for describing the relationship between logical interfaces within an entity. The ifStackTable cannot represent the full capability of this standard, since this standard makes a distinction between VLAN bindings on *ingress* to and *egress* from a Port. These relationships may or may not be symmetrical; whereas, Interface MIB Evolution assumes a symmetrical binding for transmit and receive. This makes it necessary to define other manageable objects for configuring which Ports are in the member set for which VIDs.

A PIP supports stacking through the ifStackTable. For external PIP interfaces, a PIP can be stacked on an Ethernet interface (ifType 6). If the PIP is internal, it can be stacked on an I-LAN interface to create a connection between an I-component and a B-component. The following is an example of how this process is completed:

- 1) Create I-component 1
- 2) Create VIP 1 on component 1
- 3) Create a PIP identified by ifIndex 1
- 4) Associate VIP 1 on component 1 to the PIP
- 5) Create an I-LAN interface identified with ifIndex 2
- 6) Stack PIP 1 on the I-LAN interface 2
- 7) Create B-component 2
- 8) Create CBP 1 on component 2
- 9) Associate CBP 1 on component 2 to I-LAN interface 2

The result of the previous configuration steps are displayed in Figure 17-2.

As a result, traffic can now be forwarded between I-component 1 and B-component 2 once a service is provisioned.

²⁸IANA ifType registry is <http://www.iana.org/assignments/ianafiftype-mib>.

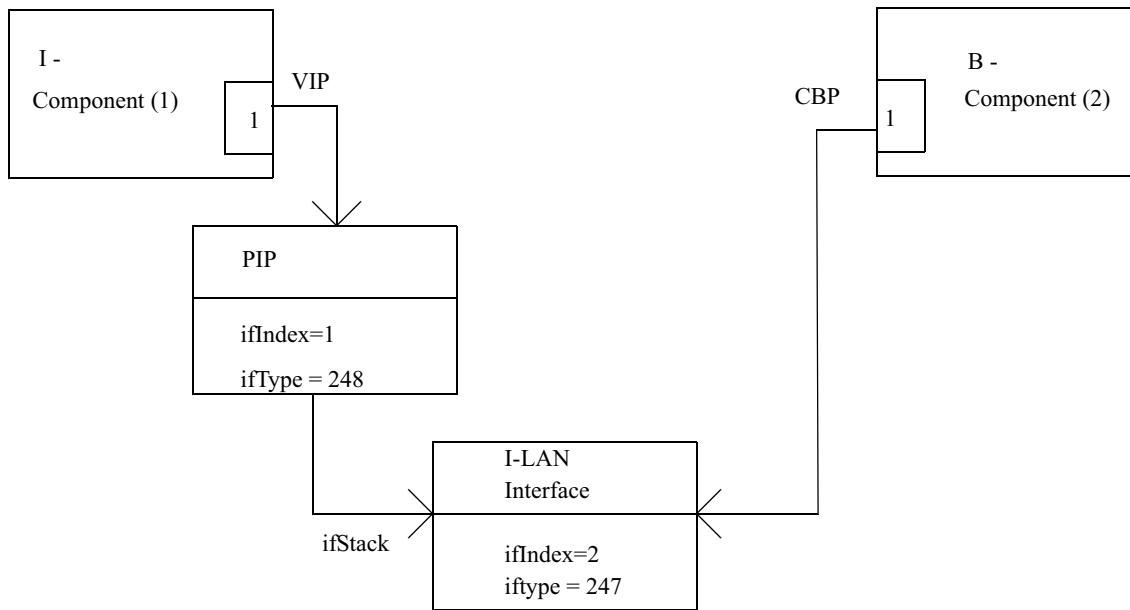


Figure 17-2—I/B-component internal LAN managed system

17.3.9 Relationship of the IEEE8021-DDCFM to other MIB modules

Subclause 17.7.9 defines the DDCFM MIB module that supports 12.17. A system implementing the DDCFM MIB module in 17.7.9 shall also implement at least the System Group of the SNMPv2-MIB defined in IETF RFC 3418 and Interfaces Group (the Interfaces MIB module, or IF-MIB) defined in IETF RFC 2863.

17.3.10 Relationship of the IEEE8021-PBBTE MIB to other MIB modules

The IEEE8021-PBBTE MIB is used to manage IB type Backbone Edge Bridges that support additional PBB-TE functionality for the configuration of TESIs across provider backbone networks. The IEEE8021-PBBTE-MIB uses textual conventions and objects from the following MIB modules:

- IEEE8021-TC-MIB
- IEEE8021-BRIDGE-MIB
- Q-BRIDGE-MIB
- IEEE8021-Q-BRIDGE-MIB

The IEEE8021-PBBTE-MIB has the purpose of managing PBB-TE specific functionality. As a PBB-TE IB-BEB is a specialized PBB IB-BEB with additional functionality, management of the system will also require that the system support the following Clause 17 MIB modules:

- IEEE8021-BRIDGE-MIB
- IEEE8021-CFM-MIB
- IEEE8021-CFM-V2-MIB
- IEEE8021-Q-BRIDGE-MIB
- IEEE8021-MSTP-MIB
- IEEE8021-PB-MIB

IEEE8021-PBB-MIB

Specifically, support of the IEEE-PBBTE-MIB requires that the system support the MIB and compliances in Table 17-21.

Table 17-21—PBB-TE required MIB compliances

MIB	Compliance
IEEE8021-PBB-MIB	ieee8021PbbWithPbbTeCompliance
IEEE8021-CFM-MIB	dot1agCfmWithPbbTeCompliance
IEEE8021-CFM-V2-MIB	

17.3.11 Relationship of the TPMR MIB to other MIB modules

Subclause 17.7.11 defines a TPMR MIB module that supports the managed objects defined in 12.19. A system implementing the TPMR MIB module in 17.7.11 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. See Table 17-17 for the key objects from the System and Interfaces groups.

17.3.12 Relationship of the IEEE8021-FQTSS MIB to other MIB modules

The IEEE8021-FQTSS MIB provides objects that extend the core management functionality of a Bridge, as defined by the IEEE8021-BRIDGE-MIB (17.7.2), in order to support the additional management functionality needed when the forwarding and queuing for time-sensitive streams extensions, as defined in Clause 34, are supported by the Bridge. As support of the objects defined in the IEEE8021-FQTSS MIB also requires support of the IEEE8021-BRIDGE-MIB, the provisions of 17.3.2 apply to implementations claiming support of the IEEE8021-FQTSS MIB.

17.3.13 Relationship of the Congestion Notification MIB to other MIB modules**17.3.13.1 Interface MIB**

Subclause 17.7.13 defines a Congestion Notification MIB (CCF MIB) module that supports congestion notification with the textual conventions imported from the TC MIB in 17.7.13. A system implementing the CCF MIB module in 17.7.13 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 other clauses that define the MIB modules in this standard. Even if a system supports none of these, if it supports the CCF MIB module, and hence, the Interfaces Group, the clarifications from the other clauses shall be applied to the Interfaces Group. The relationship between IETF RFC 2863 and IETF RFC 3418 interfaces and ports is also described in previous subclauses of 17.3. In addition, if both the CCF MIB module (17.7.13) and IEEE Std 802.3, Clause 43, are supported, a bridge component or end station may:

- 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 can be the same as the interface identifying the port. It shall not be the same as any of the IEEE 802.3 ports being aggregated.

17.3.13.2 IEEE8021-CFM and IEEE8021-CFM-V2 MIBs

Subclause 31.1 allows a system that supports either the dot1agCfmVlanTable from the IEEE8021-CFM MIB module or the ieee8021CfmVlanTable from the IEEE8021-CFM-V2 MIB module to use those tables to select the *vlan_identifier* for a transmitted Congestion Notification Message (CNM).

17.3.14 Relationship of the IEEE8021-SRP MIB to other MIB modules

The IEEE8021-SRP MIB provides objects that extend the core management functionality of a Bridge, as defined by the IEEE8021-BRIDGE MIB (17.7.2), in order to support the management functionality needed when the Stream Reservation Protocol extensions, as defined in Clause 35, are supported by the Bridge. As support of these objects defined in the IEEE8021-SRP MIB also requires support of the IEEE8021-TC-MIB, IEEE8021-BRIDGE-MIB and IEEE8021-FQTSS-MIB, the provisions of 17.3.2 apply to implementations claiming support of the IEEE8021-SRP MIB.

17.4 Security considerations

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), 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 these MIB module.

It is recommended that implementers consider the security features as provided by the SNMPv3 framework (see IETF RFC 3410 (2002), section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is not recommended. Instead, it is recommended to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of these MIB modules is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

17.4.1 Security considerations of the IEEE8021-TC MIB

This module does not define any management objects. Instead, it defines a set of textual conventions that may be used by other MIB modules to define management objects. Meaningful security considerations are contained in the following subclauses that describe security considerations for other MIB modules that define management objects with a SYNTAX of any of these textual conventions.

17.4.2 Security considerations of the IEEE8021-BRIDGE MIB

There are a number of management objects defined in the IEEE8021-BRIDGE MIB module that have 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 nonsecure environment without proper protection can have a negative effect on network operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to

control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

The following tables and objects in the IEEE8021-BRIDGE MIB can be manipulated to interfere with the operation of priority classes. This could, for example, be used to force a reinitialization of state machines, thus causing network instability. Another possibility would be for an attacker to override established policy on Port priorities, thus giving a user (or an attacker) unauthorized preferential treatment.

```
ieee8021TrafficClassesEnabled
ieee8021GmrvStatus
ieee8021PortPriorityTable
ieee8021UserPriorityRegenTable
ieee8021TrafficClassTable
ieee8021PortGarpTable
ieee8021PortGmrvTable
```

- a) The writable object ieee8021BridgeTpAgingTime controls how fast dynamically-learned forwarding information is aged out. Setting this object to a large value may simplify filtering database overflow attacks. Setting this object to too small a value may compromise the throughput of the network by causing excessive flooding.
- b) The writable ieee8021BridgeStaticTable provides a filtering mechanism controlling to which Ports frames originating from a specific source may be forwarded. Write access to this table can be used to turn provisioned filtering off or to add filters to prevent rightful use of the network.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These tables and objects and their sensitivity/vulnerability are described as follows:

- The objects ieee8021DeviceCapabilities and ieee8021PortCapabilitiesTable in the IEEE8021-P-BRIDGE-MIB could be used by an attacker to determine which attacks might be useful to attempt against a given device.
- The readable objects defined in the IEEE8021-BRIDGE MIB module provide information about the topology of a bridged network and the attached active stations. The addresses listed in the ieee8021BridgeTpFdbTable usually reveal information about the manufacturer of the MAC hardware, which can be useful information for mounting other specific attacks.
- The two notifications, newRoot and topologyChange, are emitted during spanning tree computation and may trigger management systems to inspect the status of Bridges and to recompute internal topology information. Hence, forged notifications may cause management systems to perform unnecessary computations and to generate additional SNMP traffic directed to the Bridges in a network. Therefore, forged notifications may be part of a denial of service attack.

17.4.3 Security considerations of the IEEE8021-SPANNING-TREE MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described below.

Writable objects that could be misused to cause network delays and spanning tree instabilities include:

```

ieee8021SpanningTreeVersion
ieee8021SpanningTreeRstpTxHoldCount
ieee8021SpanningTreeRstpPortProtocolMigration
ieee8021SpanningTreeRstpPortAdminEdgePort
ieee8021SpanningTreeRstpPortAdminPathCost
ieee8021SpanningTreeRstpPortAdminPointToPoint

```

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These tables and objects and their sensitivity/vulnerability are described below:

ieee8021SpanningTreeVersion could be read by an attacker to identify environments containing applications or protocols that are potentially sensitive to RSTP mode.

The writable objects

```

ieee8021SpanningTreePriority
ieee8021SpanningTreeBridgeMaxAge
ieee8021SpanningTreeBridgeHelloTime
ieee8021SpanningTreeBridgeForwardDelay
ieee8021SpanningTreePortPriority
ieee8021SpanningTreePortEnable
ieee8021SpanningTreePortPathCost

```

influence the spanning tree protocol. Unauthorized write access to these objects can cause the spanning tree protocol to compute other default topologies or it can change the speed in which the spanning tree protocol reacts to failures.

17.4.4 Security considerations of the IEEE8021-Q-BRIDGE MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described below.

The following tables and objects in the IEEE8021-Q-BRIDGE-MIB could be manipulated to interfere with the operation of virtual LANs. This could, for example, be used to force a reinitialization of state machines to cause network instability, or to change the forwarding and filtering policies.

```

ieee8021GvrpStatus
ieee8021ForwardAllTable
ieee8021StaticUnicastTable
ieee8021StaticMulticastTable
ieee8021VlanStaticTable
ieee8021PortVlanTable
ieee8021LearningConstraintsTable
ieee8021ProtocolGroupTable
ieee8021ProtocolPortTable

```

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the

values of these objects when sending them over the network via SNMP. These tables and objects and their sensitivity/vulnerability are described below.

The following read-only tables and objects in the IEEE8021-Q-BRIDGE MIB could be used by an attacker to determine which attacks might be useful to attempt against a given device, could be used by an attacker to detect whether their attacks are being blocked or filtered, or could be used to understand the logical topology of the network.

```
ieee8021MaxVlanID
ieee8021MaxSupportedVlans
ieee8021NumVlans
ieee8021FdbTable
ieee8021TpFdbTable
ieee8021TpPortGroupTable
ieee8021VlanCurrentTable
ieee8021PortVlanStatisticsTable
```

17.4.5 Security considerations of the IEEE8021-PB MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described below.

The following tables and objects in the PB-MIB could be manipulated to interfere with the operation of virtual LANs. This could, for example, be used to force a reinitialization of state machines to cause network instability, or to change the forwarding and filtering policies.

```
ieee8021PbProviderBridgePortTable
ieee8021PbVidTranslationTable
ieee8021PbCVidRegistrationTable
ieee8021PbEdgePortTable
```

17.4.6 Security considerations of the IEEE8021-MSTP MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described below.

Writable objects that could be misused to cause network delays and spanning tree instabilities include the following:

```
ieee8021CistEnableBPDUrx
ieee8021CistEnableBPDUtx
ieee8021CistPortAdminEdgePort
ieee8021CistPortMacEnabled
ieee8021MstpPortPriority
ieee8021MstpPortPathCost
```

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to

control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These tables and objects and their sensitivity/vulnerability are described below:

`ieee8021SpanningTreeVersion` could be read by an attacker to identify environments containing applications or protocols that are potentially sensitive to RSTP mode.

17.4.7 Security considerations of the IEEE8021-CFM MIB

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 17.7.7 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 tables—`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 17.7.7 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, the “Owner” of a system (in particular, a Bridge) is defined 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.

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 nonsecure environment without proper protection can have a negative effect on network operations. Table 17-22 lists the MIB tables and objects and their sensitivity/vulnerability.

Table 17-22—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.

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 17-23.

Table 17-23—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.

17.4.8 Security considerations of the IEEE8021-PBB MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described next.

The following tables and objects in the PBB-MIB could be manipulated to interfere with the operation of Provider Backbone Bridges. This could, for example, be used to force a reinitialization of state machines to cause network instability, or changing the forwarding and filtering policies. The following are vulnerable writable objects from the IEEE8021-PBB-MIB:

- ieee8021PbbVipISid
- ieee8021PbbVipType
- ieee8021PbbPipVipMap

In addition, the following are vulnerable tables from the IEEE8021-PBB-MIB:

- ieee8021PbbCBPServiceMappingTable
- ieee8021PbbPipPriorityTable
- ieee8021PbbVipToPipMappingTable
- ieee8021PbbCbpTable

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

The following read-only tables and objects in this MIB could be used by an attacker to determine which attacks might be useful to attempt against a given device, or could be used to understand the logical topology of the network:

- ieee8021PbbBackboneEdgeBridgeAddress
- ieee8021PbbVipPipIfIndex
- ieee8021PbbISidToVipComponentId
- ieee8021PbbISidToVipPort

17.4.9 Security considerations of the IEEE8021-DDCFM MIB

There are a number of management objects defined in DDCFM 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 nonsecure environment without proper protection can have a negative effect on network operations. Table 17-24 lists the MIB tables and objects and their sensitivity/vulnerability.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. Table 17-25 lists the MIB tables and objects that are sensitive to read.

Table 17-24—Sensitive managed objects (of DDCFm): tables and notifications

Table or object	Reason for sensitivity to security considerations
ieee8021DdcfmRr	Once created and activated, ieee8021DdcfmRr can intercept selected data frames traversing through a bridge interface, and send its copy encapsulated in RFM header to a specific destination within the maintenance domain. The action can create extra traffic within the maintenance domain.
ieee8021DdcfmDr	Once created and activated, ieee8021DdcfmDr will decapsulate received SFM and send the decapsulated data frame to location specified by the data frame's destination_address. This action practically injects specific traffic into the network.
ieee8021DdcfmSFMOiginator	Once created and activated, ieee8021DdcfmSFMOiginator injects SFMs into the maintenance domain. However, if the corresponding ieee8021DdcfmDr is not activated, the received SFMs are dropped. Therefore, ieee8021DdcfmSFMOiginator is less sensitive than the corresponding ieee8021DdcfmDr.

Table 17-25—Sensitive managed objects (of DDCFm) for read

Table or object	Reason for sensitivity to security considerations
ieee8021DdcfmRFMReceiver	Once created, ieee8021DdcfmRFMReceiver can receive the RFM, which encapsulate actual data frame that could be sensitive and need to be protected. It is very important to control the GET operation on this object.

17.4.10 Security considerations of the IEEE8021-PBBTE MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described below.

The following tables and objects in the PBB-MIB could be manipulated to interfere with the operation of IB type Provider Backbone Bridges. This could, for example, be used to force a reinitialization of state machines to cause network instability, or changing the forwarding and filtering policies. The following are all the writable objects from the IEEE8021-PBB-MIB:

```

ieee8021PbbTeProtectionGroupListWorkingMA
ieee8021PbbTeProtectionGroupListProtectionMA
ieee8021PbbTeProtectionGroupListRowStatus
ieee8021PbbTeTesiComponent
ieee8021PbbTeTesiBridgePort
ieee8021PbbTeTeSidEsp
ieee8021PbbTeTeSidRowStatus
ieee8021PbbTeProtectionGroupConfigCommandAdmin
ieee8021PbbTeProtectionGroupConfigWTR
ieee8021PbbTeProtectionGroupConfigHoldOff
ieee8021PbbTeProtetcionGroupConfigNotifyEnable
ieee8021PbbTeProtectionGroupISidGroupId

```

- ieee8021PbbTeProtectionGroupISidRowStatus
- ieee8021PbbTeBridgeStaticForwardAnyUnicastTable
- ieee8021PbbTeBridgeStaticForwardAnyUnicastEgressPorts
- ieee8021PbbTeBridgeStaticForwardAnyUnicastForbiddenPorts

17.4.11 Security considerations of the TPMR MIB

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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described below.

The following object in the SNMPv2-MIB could be manipulated to interfere with the operation of a management system and its ability to recognize and manage a TPMR device.

sysName

The following object in the SNMPv2-MIB could be manipulated to interfere with the operation of a management system and its ability to recognize and manage a TPMR Port.

ifName

The following objects in the TPMR-MIB could be manipulated to interfere with the operation of MSP on a TPMR Port and, for example, be used to cause network instability.

- ieee8021TpmrMspLinkNotify
- ieee8021TpmrMspLinkNotifyWait
- ieee8021TpmrMspLinkNotifyRetry
- ieee8021TpmrMspMacNotify
- ieee8021TpmrMspMacNotifyTime
- ieee8021TpmrMspMacRecoverTime

17.4.12 Security considerations of the IEEE8021-FQTSS MIB

There are a number of management objects defined in the IEEE8021-FQTSS MIB module that have 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 nonsecure environment without proper protection can have a negative effect on network operations.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than notaccessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

The following tables and objects in the IEEE8021-FQTSS MIB can be manipulated to interfere with the operation of the forwarding and queuing mechanisms in a manner that would be detrimental to the transmission of time-sensitive streams:

- ieee8021FqtssDeltaBandwidth
- ieee8021FqtssAdminIdleSlopeMs
- ieee8021FqtssAdminIdleSlopeLs
- ieee8021FqtssTxSelectionAlgorithmID
- ieee8021FqtssPriorityRegenOverride

- a) ieee8021FqtssDeltaBandwidth can be manipulated to reduce the amount of bandwidth available to a given SR class.
- b) ieee8021FqtssAdminIdleSlopeMs and ieee8021FqtssAdminIdleSlopeLs can be manipulated to change the overall amount of bandwidth available to stream traffic on a Port, in a network where SRP is not used for stream reservations.
- c) ieee8021FqtssTxSelectionAlgorithmID can be manipulated in order to apply the wrong transmission selection algorithm to a traffic class that was being used by stream traffic.
- d) ieee8021FqtssPriorityRegenOverride can be manipulated to disrupt stream traffic within an SRP domain with non-stream traffic entering from outside the SRP domain boundary.

17.4.13 Security considerations of the Congestion Notification MIB

There are a number of management objects defined in the IEEE8021-CN-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 nonsecure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/vulnerability are described in the paragraphs that follow.

Improper manipulation of certain tables and objects can result in the misidentification of the boundaries of Congestion Notification Domains (CNDs, 30.6). Any such errors can result in the following problems:

- a) Frames not originated through an RP (31.2.2.2) can pass through CPs (31.1.1). The inability to throttle these frames via Congestion Notification Messages (CNMs, 33.3) can cause legitimate, RP-originated frames to experience unacceptably high loss rates.
- b) Frames not originating through an RP (31.2.2.2) can pass through queues in the Bridged Network that are not CPs (31.1.1), where they can encounter uncontrolled congestion. This can cause those frames to experience unacceptably high loss rates.
- c) The CN-TAG (30.5) in a data frame or CNM can be improperly discarded. This can cause that CNM, or a CNM resulting from the data frame, to be discarded by the end station that sourced the frame. As a result, that flow, and all flows passing through the same CPs, can experience unacceptably high loss rates.
- d) The CN-TAG in a data frame or CNM can be improperly retained. This can result in a failure of two end stations to communicate, because the receiving end station is unable to process frames with CN-TAGs.

The following tables and objects in the IEEE8021-CN-MIB can cause such CND boundary identification errors:

ieee8021CnGlobalMasterEnable
ieee8021CnComPriDefModeChoice
ieee8021CnComPriAdminDefenseMode
ieee8021CnComPriCreation
ieee8021CnComPriLldpInstanceChoice
ieee8021CnComPriLldpInstanceSelector
ieee8021CnComPriRowStatus
ieee8021CnPortPriDefModeChoice
ieee8021CnPortPriAdminDefenseMode
ieee8021CnPortPriLldpInstanceChoice
ieee8021CnPortPriLldpInstanceSelector

Improper manipulation of certain other objects can result in assigning a data frame or CNM to the incorrect priority. This can result in the affected data streams experiencing loss rates that are either higher or lower than expected by the network administrator. Lower than expected loss rates for one data stream can cause higher than expected loss rates for other streams. These affects are typically, but not universally, less severe

than the loss rate anomalies caused by CND boundary misidentification. The following variables can cause improper priority assignment:

```
ieee8021CnGlobalCnmTransmitPriority  
ieee8021CnComPriAlternatePriority  
ieee8021CnPortPriAlternatePriority
```

Improper manipulation of the remainder of the read-write and read-create objects can cause the congestion notification algorithm to operate in ways not intended by the network administrator. This can result in excessively high data frame loss rates and/or low link utilization rates. These variables are the following:

```
ieee8021CnCpQueueSizeSetPoint  
ieee8021CnCpFeedbackWeight  
ieee8021CnCpMinSampleBase  
ieee8021CnCpMinHeaderOctets  
ieee8021CnRpPortPriMaxRps  
ieee8021CnRpgEnable  
ieee8021CnRpgTimeReset  
ieee8021CnRpgByteReset  
ieee8021CnRpgThreshold  
ieee8021CnRpgMaxRate  
ieee8021CnRpgAiRate  
ieee8021CnRpgHaiRate  
ieee8021CnRpgGd  
ieee8021CnRpgMinDecFac  
ieee8021CnRpgMinRate
```

Unintended access to any of the readable tables or variables in the IEEE8021-CN-MIB alerts the reader that congestion notification is configured, and (for all tables and variables except those in the ieee8021CnGlobalTable and ieee8021CnCpidToInterfaceTable) on which priority value or values congestion notification is configured. This information can suggest to an attacker what applications are being run, and thus suggest application-specific attacks, or to enable the attacker to detect whether their attacks are being successful or not. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

17.4.14 Security considerations of the IEEE8021-SRP MIB

The purpose of MSRP is to create reservations for various types of data streams, including Audio/Video content. Access to the objects within the IEEE8021-SRP MIB module, whether they have MAX-ACCESS of read-write, read-create, or read-only, may reveal sensitive information in some network environments. Very serious health and safety situations could arise if MSRP was involved in configuring network resources for an emergency public safety announcement and the MSRP behavior of the bridged network was allowed to be modified unexpectedly.

With these considerations in mind it is thus important to control all types of access (including GET and/or NOTIFY) to these objects and possibly even encrypt their values when sending them over the network via SNMP.

The following tables and objects in the IEEE8021-SRP MIB can be manipulated to interfere with the operation of the stream reservation mechanisms in a manner that would be detrimental to the transmission of the associated stream data:

```
ieee8021SrpBridgeBaseMsrpEnabledStatus
```

ieee8021SrpBridgeBaseMsrpTalkerPruning
ieee8021SrpBridgeBaseMsrpMaxFanInPorts
ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize
ieee8021SrpBridgePortMsrpEnabledStatus
ieee8021SrpBridgePortSrPvid

- a) ieee8021SrpBridgeBaseMsrpEnabledStatus can be manipulated to enable or disable MSRP operations for the entire Bridge.
- b) ieee8021SrpBridgeBaseMsrpTalkerPruning can be manipulated to stop the propagation of Talker attributes if Listeners aren't configured to support Talker Pruning.
- c) ieee8021SrpBridgeBaseMsrpMaxFanInPorts can be manipulated to set the number of ingress ports supporting streaming down to one, which would stop Talkers streams from coming in on any other port.
- d) ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize can be manipulated to set a frame size that is so large or so small that it causes the Bridge to calculate unreasonable maximum latency.
- e) ieee8021SrpBridgePortMsrpEnabledStatus can be manipulated to enable or disable MSRP on a particular port, perhaps allowing sensitive stream data to be sent to unacceptable devices.
- f) ieee8021SrpBridgePortSrPvid can be manipulated to move Streams to a VLAN that has been blocked by management, thus disabling reception of the Stream by one or more Listeners.

17.5 Dynamic component and Port creation

17.5.1 Overview of the dynamically created Bridge entities

The IEEE Bridge MIBs allow for the possibility of Bridge components and Bridge Ports to be created under the control of management operations. This functionality is optional. Compliant Bridges need not support this functionality or may tie the creation of components and Ports to the insertion or extraction of system hardware. This subclause outlines the rules and table interactions used by management stations to create soft configurable dynamic components and Ports on those systems that support this functionality.

A Bridge component contains, at a minimum, a collection of Bridge Ports, a relay unit that forwards and filters frames that travel between Ports, and managed objects to control the operation of the component. Certain types of components may contain other objects. These will be addressed later in this subclause in component specific text.

Bridge components are the Owners of Bridge Ports, so components must be created before components can be populated with Ports.

The tables that are indexed by a single component ID index are as follows:

ieee8021BridgeBaseTable
ieee8021QBridgeTable
ieee8021QBridgeNextFreeLocalVlanTable
ieee8021QBridgeLearningConstraintDefaultTable
ieee8021CistTable
ieee8021MstConfigIdTable
ieee8021CfmVlanTable

17.5.1.1 Components

A component contains a relay function whose purpose is to move frames between interfaces to the relay called Bridge Ports. Different types of components are provided in IEEE Std 802.1Q that are used to construct different types of Bridges.

17.5.1.2 Bridge Ports

A Bridge Port is a frame source or sink directly attached to the relay function of a Bridge component.

17.5.1.3 Internal LAN connections

A Backbone Edge Bridge (BEB) is composed of zero or one B-component and zero or more I-components. When the BEB has both a B-component and some I-components, the PIPs and CBPs of these I-components are connected by internal LAN connections. There needs to be a way to specify the interconnections between the PIPs on the I-component and the Customer Backbone Port on the B-component. This is done via the ieee8021BridgeILanIfTable defined in the IEEE8021-BRIDGE MIB.

Essentially, this table allows for the creation of a new “interface” to represent the connection and then the ifStackTable is used to specify the interconnection.

These interconnections are used in multi-component Bridges, such as Provider Edge Bridges and BEBs, to specify the relationship between two interfaces in the following manner:

Two Port interfaces are interconnected if the invocation of a request operation at one of the interfaces causes an indication operation with the same parameters to happen at the other interface.

17.5.1.4 Provider Instance Ports

Even though it is not a Bridge Port, the creation of Provider Instance Ports on I-components are discussed as part of the Bridge Port creation logic for I-components.

17.5.2 Component creation

A component is created by making an entry in the ieee8021BridgeBaseTable with the ieee8021BridgeBaseComponentType set to the proper value.

A Bridge component consists of a relay function and related Bridge Ports. The component type determines if the relay operates on untagged, C-tagged, or S-tagged frames. It also determines which specific type(s) of Bridge Ports may be created on the component.

17.5.2.1 D-component creation

The D-component has no specific component creation rules.

17.5.2.2 C-VLAN component creation

C-VLAN components are used in two different types of Bridges. The first is the C-VLAN component of a customer VLAN Bridge. The second is as the C-VLAN component of a Provider Edge Bridge.

Provider Edge Bridge C-VLAN components are created implicitly by the creation of a Customer Edge Port on the S-VLAN component of the Provider Edge Bridge. C-VLAN components that belong to customer Bridges are created by a management station performing a row-create on the component table or by implicit action such as the insertion of blades into a system.

17.5.2.3 S-VLAN component creation

The S-VLAN component has no specific component creation rules.

17.5.2.4 B-component creation

The B-component has no specific component creation rules.

17.5.2.5 I-component creation

The I-component has no specific component creation rules.

17.5.3 Port creation

This subclause discusses how Ports of each relevant Port type are created on each relevant component type.

The general procedure is for the network administrator to perform an SNMP row-create operation on a table specific to the type of Port being created. If the operation succeeds, an entry will be implicitly created in the ieee8021BridgeBasePortTable by the agent.

The specific details are outlined in the 17.5.3.1 through 17.5.3.5.

17.5.3.1 Port creation on D-components

D-components are used in IEEE 802.1D Bridges. Creation of a Port on a D-component requires specifying the component and Port index in the ieee8021BridgeCreatableBaseBridgeGroup table.

The type of the component referred to by the component ID parameter must be dBridgeComponent.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBaseComponentId	- As per IEEE 802.1D Port Table
ieee8021BridgeBasePort	- As per IEEE 802.1D Port Table
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilites	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit customerVlanPort(0) must be set
ieee8021BridgeBasePortExternal	- dBridgePort(8)
	- Implementation Specific

17.5.3.2 Port creation on C-VLAN components

C-VLAN components support three different types of Bridge Ports. These are Customer VLAN Ports, Customer Edge Ports, and Provider Edge Ports.

A C-VLAN component that is not part of a Provider Edge Bridge may have Customer VLAN Ports. A C-VLAN component that is part of a Provider Edge Bridge must have exactly 1 Customer Edge Port and any number of Provider Edge Ports.

The only type of Ports that may be created by operating on the C-VLAN component are the Customer VLAN Ports.

Customer Edge Ports are created by a management action on the S-VLAN component of a Provider Edge Bridge. From a management perspective, these entities are managed through the S-VLAN component or via management operations specific to the Provider Edge Bridge.

Provider Edge Ports, together with the associated CNP and the internal connection between them, are created by performing row-create operations for a C-VID in the ieee8021PbCvidRegistrationTable and the corresponding S-VID in the ieee8021PbEdgePortTable.

17.5.3.2.1 Creating Customer VLAN Ports

Customer VLAN Ports are created by performing a row-create operation on the ieee8021QBridgeCVlanPortTable for a C-VLAN component that is configured to act as an IEEE 802.1Q Bridge. The required columns are the component ID and the Port Number to use for the newly created Port.

The type of the component referred to by the component ID parameter must be cVlanComponent configured for Q-Bridge operation.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per QBridgeCVlanPortTable
ieee8021BridgeBasePort	- As per QBridgeCVlanPortTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilites	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit customerVlanPort(0) must be set
ieee8021BridgeBasePortExternal	- customerVlanPort(2)
	- Implementation Specific

17.5.3.2.2 Creating CEPs

CEPs are created by doing a RowStatus create operation on the CEP table belonging to the PEB.

17.5.3.2.3 Creating PEPs

Provider Edge Ports, together with the associated CNP and the internal connection between them, are created by performing row-create operations for a C-VID in the ieee8021PbCvidRegistrationTable and the corresponding S-VID in the ieee8021PbEdgePortTable. See the discussion on provisioning a C-tagged service interface in 17.6.1.2.

17.5.3.3 Port creation on S-VLAN components

S-VLAN components are the S-VLAN component of an S-VLAN Bridge or Provider Bridges. There are two different types of Ports that may be created on S-VLAN components. These are PNPs and CNPs.

17.5.3.3.1 Creating PNPs

A PNP on an S-VLAN component works pretty much the same way as a customerVlanPort on a C-VLAN component except that the VID value used for forwarding and filtering decisions must come from an S-TAG and not a C-TAG.

Creating a PNP on a PB Bridge is done by adding an entry to the ieee8021PbPnpTable specifying the componentId and Port Number.

ieee8021BridgeBasePortComponentId	- Component to which the new Port belongs
ieee8021BrdigeBasePort	- Port Number for the new Port

The type of the component referred to by the component ID parameter must be sVlanComponent.

This will cause an implicit entry in the ieee8021QBridgePortVlan table associated with the S-VLAN component.

The Component ID must refer to the S-VLAN component of a Provider Edge Bridge and the Port Number must refer to the Port Number of the Provider Network Port associated with that S-VLAN component.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per PbPnpTable
ieee8021BridgeBasePort	- As per PbPnpTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit providerNetworkPort(1) must be set
ieee8021BridgeBasePortExternal	- providerNetworkPort(3)
	- Implementation Specific

17.5.3.3.2 Creating CNPs

There are two variants on the creation of CNPs. CNPs are either internal or external. Internal Ports are directly connected to a PEP on the C-Vlan component of a PEB and provide a C-tagged service interface. External Ports are connected to an external customer system and provide either a Port-based or S-tagged service interface.

a) Creating the CNP external variant

The external variant of a CNP is used to provide a Port-based or S-tagged service interface to customer of the Provider Bridged Network.

Creating a CNP of the external variant requires specifying the B-component ID and Bridge Port Number of the Port to be created. This is done by creating an entry in the ieee8021PbCnpTable.

This creates a new entry in the ieee8021QBridgePortVlanTable for the provider Bridge's S-VLAN component.

This operation also created a new entry in the ieee8021BridgeBasePortTable.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per PbCnpTable
ieee8021BridgeBasePort	- As per PbCnpTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit customerNetworkPort(2) must be set
ieee8021BridgeBasePortExternal	- customerNetworkPort(4)
	- Implementation Specific

b) Creating the CNP internal variant

The internal variant of a CNP is used to provide a C-tagged service interface to the customer of a Provider Bridged Network.

The CNP, internal variant, together with the associated PEP and the internal connection between them, are created by performing row-create operations for an S-VID in the ieee8021PbEdgePortTable and the corresponding C-VID in the ieee8021PbCvidRegistrationTable. See the discussion on provisioning a C-tagged service interface in 17.6.1.2.

Internal CNPs are not directly manageable.

17.5.3.3 Creating a CEP

CEPs are created by doing a row-create operation the Provider Edge Bridge's CEP table. The ieee8021PbCepTable contains the following columns:

ieee8021BridgeBasePortComponentId	- S-VLAN component ID that “owns” the CEP
ieee8021BridgeBasePort	- The Port Number of the CEP on the S-VLAN component.
ieee8021PbCepCComponentId	- C-VLAN component read only index cross-ref for entity MIB
ieee8021PbCepCepPortNumber	- C-VLAN component Port read-only index cross-ref for entity MIB
ieee8021PbCepRowStatus	- Controls the creation and deletion of the Port

Note that the C-VLAN component containing the newly created CEP does not appear in the IEEE 802.1 Bridge MIB’s list of components. So ieee8021PbCepCComponentId and the ieee8021PbCepCepPortNumber, are index values that are not further interpreted by any IEEE 802.1 MIB. These index values, if present, can be used in an implementation-dependent manner to allow management stations to cross reference entries in other MIBs, such as the IETF’s entity MIB, back to the information managed by the IEEE MIBs.

The newly created Port will be added to the Port list of the S-VLAN component of the Provider Edge Bridge.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per CustomerEdgePortTable
ieee8021BridgeBasePort	- As per CustomerEdgePortTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit customerEdgePort(3) must be set
ieee8021BridgeBasePortExternal	- customerEdgePort(5)
	- Implementation Specific

17.5.3.4 Port creation on B-components

B- and I-components are the two components that constitute a Backbone Edge Bridge (see Clause 26). These components provide what is generally known as “MAC in MAC” transport of Ethernet frames.

Ports on these BEB components are created using 12.16 managed objects.

17.5.3.4.1 Creating PNPs

A B-component is an S-VLAN component with one or more Customer Backbone Ports. A B-component may be a component of a Backbone Edge Bridge. PNP creation on a B-component follows the same logic used to create a PNP on an S-VLAN component.

Creating a PNP on a BEB is done by adding an entry to the ieee8021PbPnpTable specifying the componentId and Port Number.

ieee8021BridgeBasePortComponentId	- Component to which the new Port belongs
ieee8021BridgeBasePort	- Port Number for the new Port

The type of the component referred to by the component ID parameter must be sVlanComponent.

As a PNP is a Port on a B-component, which is a type of S-VLAN component, the act of creating a PNP causes entries to be made in the ieee8021QBridgePortVlanTable and the ieee8021BridgeBasePortTable.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per PbPnpTable
ieee8021BridgeBasePort	- As per PbPnpTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific bit providerNetworkPort(1) must be set
ieee8021BridgeBasePortType	- providerNetworkPort(3)
ieee8021BridgeBasePortExternal	- Implementation Specific

17.5.3.4.2 Creating CBPs

Creating a CBP on a B-component is done by creating an entry in the ieee8021PbbCbpTable with the following column values:

ieee8021BridgeBasePortComponentId	- Component number of the B-component
ieee8021BridgeBasePort	- Port Number for the newly created Port

As a CBP is a Port on a B-component, which is a type of S-VLAN component, the act of creating a PNP causes entries to be made in the ieee8021QBridgePortVlanTable and the ieee8021BridgeBasePortTable.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per PbbCbpTable
ieee8021BridgeBasePort	- As per PbbCbpTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific bit customerBackbonePort(4) must be set
ieee8021BridgeBasePortType	- customerBackbonePort(6)
ieee8021BridgeBasePortExternal	- Implementation Specific

17.5.3.5 Port creation on I-components

Creating a Bridge Port on an I-component works the same way as creating a Port on a B-component with the exception that the rules for determining the legality of the Port type are different.

Creating PIPs works in a manner analogous to the creation of a Bridge Port, but as PIPs are not Bridge Ports the details are somewhat different.

17.5.3.5.1 Creating CNPs

The CNP is used to provide a Port-based or S-tagged service interface to customer of the Provider Backbone Bridged Network.

Creating a CNP requires specifying the I-component ID and Bridge Port Number of the Port to be created. This is done by creating an entry in the ieee8021PbCnpTable.

This creates a new entry in the ieee8021QBridgePortVlanTable for the provider Bridge's I-component.

This operation also created a new entry in the ieee8021BridgeBasePortTable.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per QBridgePortVlanTable
ieee8021BridgeBasePort	- As per QBridgePortVlanTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit (2) must be set
ieee8021BridgeBasePortExternal	- customerNetworkPort(4)
	- Implementation Specific

17.5.3.5.2 Creating VIPs

The creation of VIP starts with the creation of an entry in the ieee8021PbbVipTable with the following columns set:

ieee8021BridgeBasePortComponentId	- Component Number of I-component
ieee8021BridgeBasePort	- As appropriate
ieee8021PbbVipRowStatus	- Probably either createAndGo or createAndWait

An entry will be created in the ieee8021PbbVipTable with the following columns set:

ieee8021PbbVipPipIfIndex	- 0
ieee8021PbbVipISid	- 1
ieee8021PbbVipDefaultDstBMAC	- 00-1E-83-00-00-01
ieee8021PbbVipType	- IngressAndEgress
ieee8021PbbVipRowStatus	- "Active"

As a VIP is a Port on a I-component, which is a type of S-VLAN component, the act of creating a VIP causes entries to be made in the ieee8021QBridgePortVlanTable and the ieee8021BridgeBasePortTable.

The implicitly constructed ieee8021BridgeBasePortTable entry will have the following fields filled in:

ieee8021BridgeBasePortComponentId	- As per QBridgePortVlanTable
ieee8021BridgeBasePort	- As per QBridgePortVlanTable
ieee8021BridgeBasePortIfIndex	- Implementation Specific Action
ieee8021BridgeBasePortDelayExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortMtuExceededDiscards	- Statistic, reset to 0 by creation
ieee8021BridgeBasePortCapabilities	- Implementation Specific
ieee8021BridgeBasePortTypeCapabilities	- Implementation Specific
ieee8021BridgeBasePortType	bit virtualInstancePort must be set
ieee8021BridgeBasePortExternal	- virtualInstancePort
	- Implementation Specific

Note that before becoming operational, the VIP must be assigned to a PIP and must be assigned to a service by setting an I-SID value. Furthermore, the mapping tables must be set up in a consistent basis.

17.5.3.5.3 Creating PIPs

PIPs are not Bridge Ports, but they are necessary parts of the internal operation of a BEB. They share many characteristics of Bridge Ports, simply because both PIPs and Bridge Ports are interfaces that transfer MAC frames, but are not connected to a MAC relay function.

PIPs are created using a Backbone Edge Bridge managed object. This object maps to the ieee8021PbbPipTable in the PBB MIB module. An entry needs to be created in the table with the following attributes:

ieee8021PbbPipIfIndex	- A previously unallocated ifIndex value
ieee8021PbbPipBMACAddress	- Implementation Default or specified value
ieee8021PbbPipName	- Implementation Default or specified value
ieee8021PbbPipIComponentId	- I-component index to which this PIP belongs
ieee8021PbbPipVipMap	- Empty Mapping

Note that before the operation can occur on this PIP, it may be necessary to modify the B-MAC address value of the PIP, especially in the case where local MAC addresses are used in the Provider Backbone Bridged Network.

17.5.3.6 Required post creation operations

Before a Bridge Port is ready for operational use, the Bridge Port must be associated with an entity that acts as a frame source and frame sink. For a Bridge with or without the dynamic Port creation capability, this will likely be an ISS. For a Bridge with dynamic Port creation, it will be an ISS.

Most Bridge Ports manage this association by using the ieee8021BridgeBasePortIfIndex column in the ieee8021BridgeBasePort table. It is system dependent if the ieee8021BridgeBasePortIfIndex column in the ieee8021BridgeBasePort table needs to be filled in with the appropriate ifIndex for the newly created Bridge Port. Systems that enforce a specific mapping between Bridge Port Numbers and physical interfaces may have the agent automatically fill in this field and the association may not be modifiable.

The association for Customer Edge Ports is managed by the ieee8021PbCepTable.

The association between Provider Edge Ports and Customer Network Ports for PEBs is managed implicitly by the creation of the PEP/CNP pair by adding a CEP to the member set of an S-VID.

The association between PIPs and CBPs is managed by the internal LAN table (ieee8021BridgeILanIfTable).

Additionally, for Ports on elements of Provider Bridges and Provider Backbone Bridges, instances in service tables must be created before the Ports are capable of passing frames according to the relevant clauses of IEEE Std 802.1Q and subsequent amendments.

17.6 MIB operations for service interface configuration

This subclause outlines, on a per-service interface basis, the MIB operations that a Provider must perform to configure each type of service interface. The clause is organized into two broad subclauses. The first subclause explains how service interfaces for Provider networks (Clause 15, Clause 16, etc.), are configured. The second subclause explains how service interfaces for Provider Backbone Networks (Clause 25, Clause 26, etc.), are configured. Both of these subclauses have the same substructure. First, a description of the MIB operations that are service interface type independent, and secondly a description of the service type MIB operations organized in a per-service type manner.

17.6.1 Provisioning Provider Bridged Network service interfaces

This clause assumes that the components and Ports, if dynamic, have been created, although there may be some notes on the parameters used for Port creation on a per-service basis, if there are restrictions. This subclause describes the MIB operations needed to configure service interfaces for Provider Bridged Network equipment.

Table 17-26 lists the parameters required for each of the service interface types that can be used to interface to a provider network. These parameter names are used in subsequent clauses as values to be entered in the appropriate MIB tables.

Table 17-26—Provider Bridge service interface parameters

Parameter	Description	Customer service interface type		
		Port-based	C-tagged	S-tagged
ComponentID	ID of the S-VLAN component	Mandatory	Mandatory	Mandatory
Port Number	Bridge Port Number that Receives Customer Frames	CNP Port Number	CEP Port Number	CNP Port Number
RelaySVid	S-VID value used within the provider network	Mandatory	Mandatory	Mandatory
CVid	C-VID value used to identify frames to be transported by the specified service	Not needed	Mandatory	Not needed
LocalSVid	S-VID value used to identify the frames to be transported by the service	Not needed	Not Needed	Optional If not specified, the RelaySVid value is used
Default CVID value	C-VID value to be used for frames received from the Provider Network without C-Tags	Not needed	Mandatory	Not needed
Default Priority Value	Priority value to be used for frames received from the Provider Network without C-Tags	Not needed	Mandatory	Not needed

17.6.1.1 Provisioning a Port-based service interface

This subclause describes the MIB operations needed to provision a Port-based service interface.

Creating a Port-based service on a CNP requires manipulation of the Port member sets of the ieee8021QBridgeVlanStaticTable. One must manipulate the Port member sets for the row with the following index values:

ieee8021QBridgeVlanStaticComponentId	- ComponentId
ieee8021QBridgeVlanStaticVlanIndex	- RelaySVid

The CNP must be added to the PortList specified by the ieee8021QBridgeVlanStaticEgressPorts column of this row and must be added to the PortList specified by the ieee8021QBridgeVlanStaticUntaggedPorts column of this row.

One may, or may not, wish to allow the customer to specify priorities for the arriving customer traffic. In either case one must set the ieee8021QBridgePortAcceptableFrameTypes column of the entry in the ieee8021QBrigePortVlanTable indexed by

ieee8021BridgeBasePortComponentId	- ComponentId
ieee8021BridgeBasePort	- PortNumber

to admitUntaggedAndPriority and set the ieee8021QBridgePortIngressFiltering column to true to ensure that customers may not “spoof” the provider network by sending S-tagged frames to the CNP.

17.6.1.2 Provisioning a C-tagged service interface

This subclause describes the MIB operations needed to provision a C-tagged service interface.

The Customer Edge Port must be added to the Port member set of each S-VID on the S-VLAN component of the Provider Edge Bridge that is to provide service for the CEP. This is done by modifying the ieee8021QBridgeVlanStaticTable ieee8021QBridgeStaticEgressPorts PortList whose matching index column values are as follows:

ieee8021QBridgeVlanStaticComponentId	- ComponentId
ieee8021QBridgeVlanStaticVlanIndex	- RelaySVid

Adding the CEP to the member set of an S-VID implies the existence of a CNP on the S-VLAN component that is internally connected to a PEP on the C-VLAN component. Enabling the CEP to send and receive frames from the service instance identified by the RelaySVid requires creating a linkage between the S-VID, the CNP, the PEP, and a C-VID. This is accomplished by configuring the ieee8021PbEdgePortTable and ieee8021PbCVIDRegistrationTable.

Each S-VID value that is associated with a provider service instance accessed through the CEP requires that an entry be made in the ieee8021PbEdgePortTable. An entry in this table maps an S-VID to a CNP and associates that CNP with a specific PEP. The CNP is identified by the CEP port number and the S-VID that form the index values for the table. The contents of the table entry identify the PEP by specifying the C-VID value used as the PVID for that PEP (as well as providing other configuration parameters for that PEP). A CNP is implicitly created by the first entry identifying a given PEP (i.e. the first entry that contains a given C-VID value). In some cases it is useful to create multiple entries to map multiple S-VIDs to the same CNP. Two or more entries map to the same CNP if the C-VID value identifying the PVID of the PEP is the same (the other values of the entry should be the same as well). This many-to-one mapping of S-VIDs to CNP/PEP pairs can be used to configure asymmetric VLANs for supporting rooted multipoint connectivity

(F.1.3.2). The table entry with the following index values:

ieee8021BridgeBasePortComponentId	- ComponentId
ieee8021BridgeBasePort	- PortNumber
ieee8021PAbEdgePortSvid	- RelaySvid

needs to have the following columns set:

ieee8021PbEdgePortPVID	- DefaultCvid
ieee8021PbEdgePortDefaultUserPriority	- DefaultPriority
ieee8021PbEdgePortAcceptableFrameType	- As Desired
ieee8021PbEdgePortEnableIngressFiltering	- As Desired

Each C-VID value that is to be mapped to a provider service instance requires that an entry be made in the ieee8021PbCvidRegistrationTable. An entry in this table maps a C-VID to a PEP and associates that PEP with a specific CNP. The PEP is identified by the CEP port number and the C-VID that form the index values for the table. The contents of the table entry identify the CNP by specifying the S-VID value used as the PVID for that CNP (as well as providing member set and untagged set information for the C-VID). A PEP is implicitly created by the first entry identifying a given CNP (i.e. the first entry that contains a given S-VID value). In some cases it is useful to create multiple entries to map multiple C-VIDs to the same PEP. Two or more entries map to the same PEP if the S-VID value identifying the PVID of the CNP is the same. This many-to-one mapping of C-VIDs to PEP/CNP pairs can be used to support bundling multiple C-VLANs into a single service instance. The table entry with the following index values:

ieee8021BridgeBasePortComponentId	- ComponentId
ieee8021BridgeBasePort	- PortNumber
ieee8021PbCvidRegistrationCvid	- Cvid

needs to have the following columns set:

ieee8021PbCvidRegistrationSvid	- RelaySvid
ieee8021PbCvidRegistrationUntaggedPep	- As Desired
ieee8021PbCvidRegistrationUntaggedCep	- As Desired

17.6.1.3 Provisioning an S-tagged service interface

This subclause describes the MIB operations needed to provision an S-tagged service interface.

Creating an S-tagged based service interface requires manipulation of the Port member sets of the CNP entry in the ieee8021QBridgeVlanStaticTable. One must manipulate the Port member sets for the row with the following index values:

ieee8021QBridgeVlanStaticComponentId	- ComponentId
ieee8021QBridgeVlanStaticVlanIndex	- RelaySvid

The CNP must be added to the PortList specified by the ieee8021QBridgeVlanStaticEgressPorts column of this row. The CNP must not be in the PortList specified by the ieee8021QBridgeVlanStaticUntaggedPorts column of this row as this service requires S-TAGS in all frames.

If the Provider wishes to use separate S-TAG values for the customer and provider S-VLAN spaces, then the entry in the ieee8021PbVidTranslationTable indexed by the following:

ieee8021BridgeBasePortComponentId	- ComponentId
ieee8021bridgeBasePort	- PortNumber

ieee8021PbVidTranslationLocalVid	- LocalSVid
----------------------------------	-------------

must be created or changed so that the ieee8021PvVidTranslationRelayVid column is set to the RelaySVid.

17.6.2 Provisioning Backbone Bridged Network service interfaces

This subclause describes the MIB operations needed to provision service interfaces on Provider Backbone Bridged Networks. In particular, this subclause assumes that the provider backbone network service instance exists and that the following information is available:

The I-SID value that identifies the correct backbone service instance is known.

At any BEB that is to have a service interface map customer data to a backbone service instance, given the I-SID, the following information can be determined:

- The Component ID of the B-component that contains the appropriate PNP used to access the service instance.
- The Port number of the PNP.
- The B-VID value used to transport that service over the Backbone Core.
- The DA used to transport the frame over the Backbone Core. Either a default MAC address, or the service MAC address.

Furthermore, the core of the network is assumed to be operationally correct. That is, if a frame is presented to the Backbone Core by a BEB with the proper B-VID value and proper B-DA at the proper PNP, then the frame will arrive at the appropriate VIP(s). This can be achieved by statically configuring the member sets of the Backbone Core Bridges or by configuring the network such that dynamic address and VLAN registration protocols ensure that the Core Bridges Port member sets and filtering databases are set appropriately.

Provisioning a service interface in a Backbone Edge Bridge to access a backbone service interface requires configuring two components. The first is the I-component. The I-component determines the backbone service instance to which customer data is mapped. The second component is the B-component. The B-component determines how the backbone service instance is forwarded over the PBBN core. Thus, the I-component determines the I-SID value, and thus the backbone service instance, to which the customer data belongs and the B-component determines the B-VID and B-DA values that will be used by the relay functions of the Backbone Core Bridges to determine the forwarding path through the core network used by the customer's data.

In the case of an I-tagged service interface, the service interface used by the customer is configured on the CBP of a Backbone Edge Bridge and the physical interconnection between the backbone network and the customer network is between the CBP of a provider managed B-component and the PIP of a customer managed I-component.

To simplify the discussion of service interface provisioning, this subclause first discusses interface type independent provisioning of the B-component, and then service interface independent provisioning of the I-component. Following that discussion, interface type specific configuration is discussed. This type specific configuration consists of instructions on how to determine the parameters needed by the service independent configuration of both the I- and B-components along with the provisioning of service specific tables in the I- and B-components.

Table 17-27 lists the parameters required to provision each type of service interface per Clause 25. The parameters are used in the descriptions of the MIB table operations needed to configure each service in the following subclauses.

Table 17-27—Provider Backbone Bridge service interface parameters

Parameter	Description	Service interface type			
		Port-based	Unbundled one-to-one S-tagged	Bundled many-to-one or all-to-one S-tagged	I-tagged service interface
IComponentID	Component ID of the I-component of a Backbone Edge Bridge	Mandatory	Mandatory	Mandatory	Not needed
BComponentID	Component ID of the B-component of a Backbone Edge Bridge	Mandatory	Mandatory	Mandatory	Mandatory
CBPPortNumber	Port number of a Customer Backbone Port	Mandatory	Mandatory	Mandatory	Mandatory
BackboneSid	I-SID value used between the PIP and CBP	Mandatory	Mandatory	Mandatory	Mandatory
MappingType	Used for point-to-multipoint services where ingress or egress limiting is needed	Optional	Optional	Optional	Optional
DefaultDestAddress	MAC address to use as the backbone DA	Optional. If not specified, the service default address is used.	Optional. If not specified, the service default address is used.	Optional. If not specified, the service default address is used.	Optional. If not specified, the service default address is used.
B-Vid	Backbone VID that identifies the VLAN to transport the service over the backbone	Optional. If not specified, the default B-VLAN is used	Optional. If not specified, the default B-VLAN is used	Optional. If not specified, the default B-VLAN is used	Optional. If not specified, the default B-VLAN is used
LocalSID	I-SID value to be used in the backbone network.	Optional. If not specified, the BackboneSID value is used	Optional. If not specified, the BackboneSID value is used	Optional. If not specified, the BackboneSID value is used	Optional. If not specified, the BackboneSID value is used
PCPSpec	Priority code point encoding and decoding specifications	Optional. Note that separate values may be required for the I- and B-components	Optional. Note that separate values may be required for the I- and B-components	Optional. Note that separate values may be required for the I- and B-components	Not needed

Table 17-27—Provider Backbone Bridge service interface parameters (continued)

Parameter	Description	Service interface type			
		Port-based	Unbundled one-to-one S-tagged	Bundled many-to-one or all-to-one S-tagged	I-tagged service interface
LocalSVid	S-TAG value that designates traffic at the CNP, if different from the RelaySVid	Not needed	Optional. RelaySVid used if not specified	Optional. RelaySVid used if not specified	Not needed
RelaySVid	S-TAG value used within the I-component to identify traffic belonging to a particular service instance	Mandatory	Mandatory	Mandatory	Not needed
VIPPortNumber	Port number of the VIP	Mandatory	Mandatory	Mandatory	Not needed
PIPIndex	Identifier of the PIP to which the VIP belongs	Mandatory	Mandatory	Mandatory	Not needed
VIP-SID	ISID value used by a VIP	Mandatory	Mandatory	Mandatory	Not needed
adminPoint-ToPointMAC	Indicates if the service is a point-to-point service	Optional	Optional	Optional	Optional
CNPPortNumber	The Bridge Port number of a CNP on an I-component	Mandatory	Mandatory	Mandatory	Not needed

17.6.2.1 Service type independent provisioning of the B-component

All frames that are transmitted over a Provider Backbone Bridged Network service instance ultimately start at a Customer Backbone Port (CBP). Thus part of provisioning any service interface requires configuration of the CBP. This is done with the ieee8021PbbCBPServicesMappingTable.

Provisioning this portion of the service interface is straightforward. One needs to add a row to the ieee8021PbbCBPServicesMappingTable whose columns are set to the values specified in the required parameters. Furthermore, one may need to ensure that both the CBP and PNP belong to the member set of the B-VID if dynamic configuration of VIDS is not supported.

17.6.2.2 Service type independent provisioning of the I-component

The I-component needs to be configured when either Port-based or S-tagged service interfaces are configured. Traffic belonging to an I-tagged service interface does not transit the I-component; hence there is no configuration of the I-component for that service. The I-component's provisioning is more service interface dependent than the provisioning of the B-component. However, some of the logic needed to provision the I-component does not depend upon the service being provisioned. In particular the configuration of the ieee8021PbbVIPTable and the ieee8021PbbPipTable is provisioned almost identically for both Port-based and S-tagged service instances.

17.6.2.2.1 Configuring the ieee8021PbbVipTable

Traffic for both Port-based and S-tagged services transitions a VIP on the I-component. The particular VIP over which the traffic transits determines the backbone service instance, and thus the I-SID value for the traffic. A VIP must be created for each service instance for which the I-component forwards

traffic. Bundled services interfaces map multiple customer S-VIDs to a single service instance. Generally, when creating a service the first step is to create the VIP that will carry this service. This is done by creating a row in the ieee8021PbbVipTable. The following columns must be set in the table:

ieee8021BridgeBasePortComponentId	- set to IComponentId
ieee8021BridgeBasePort	- set to VIPPortNum
ieee8021PbbVipISid	- set to the VIP-SID (which may be localSID or backboneSID depending on the use of I-SID translation at the B-component)
ieee8021PbbVipType	- set as appropriate depending upon the symmetry of the vlan

As a side effect of this row creation, a row will be created in the ieee8021PbbISidToVipTable. Essentially, this table implements the inverse mapping, taking an I-SID value to the VIP that is the local endpoint of the service instance.

17.6.2.2 Configuring the ieee8021PbbPipTable

For Port-based, unbundled service interfaces, and for the first S-VID allocated to a bundled service, then the VIP used for the service must be configured to use the appropriate PIP. Use the IComponentID and PIPIndex to select a row in this table. Set the bit corresponding to the VIPPortNum in the appropriate MAP column of that row to 1. This adds the VIP to the PIP and, as a side effect, sets the ieee8021PbbVipPipIfindex column in the ieee8021PbbVipTable.

17.6.2.3 Service dependent provisioning for an I-tagged service interface

The I-tagged service interface of a PBBN allows a customer to attach directly to a Customer Backbone Port of a B-component of a BEB allowing the customer to transport frames that already possess an I-TAG. A scenario for this is when the customer is also a backbone provider and is purchasing transport across another Provider's network. Another scenario is when an access network provides an I-tagged interface. The I-tagged service interface is defined in 25.5

The provisioning is accomplished by using the parameter values of Table 17-7 and applying the procedures outlined in 17.6.2.1 to provision the B-component.

17.6.2.4 Service dependent provisioning a Port-based service interface

The port-based service interface of a PBBN causes all of the traffic arriving at the Customer Network Port of the I-component of a BEB to mapped to a single backbone service instance. The Port-based interface of a PBBN provides the same customer service as a Port-based service interface of a PBN. The implementation of this service interface in a PBBN is defined in 25.3.

17.6.2.4.1 Configuring the B-component

Given the PIP, one uses the I-LAN table to find the B-component and CBP that the traffic for this service will transit. This gives us the following two parameters:

BComponentId —B-component ID for the CBP
CBPPortNumber—Port number of the CBP on this B-component

These two service parameters found from the I-LAN table along with the remainder of the parameters defined in Table 17-5 are used to configure the B-component as outlined in 17.6.2.1. As a side effect of configuring the CBP via the ServiceMapping table, the ieee8021PbbVipDefaultDstMAC column is set in the ieee8021PbbVipTable.

17.6.2.4.2 Configuring the Port operating modes and I-component VID

The first step is to configure the I-component VID used to interconnect the VIP with the CNP. This is done by creating an entry in the ieee8021QBridgeVlanStaticTable with the columns set to the following values:

ieee8021QBridgeVlanStaticComponentId	- IcomponentId
ieee8021QBridgeVlanStaticVlanIndex	- IcompPVid
ieee8021QBridgeVlanStaticName	- Something administratively convenient
ieee8021QBridgeVlanStaticEgressPorts	- Port List containing just the VIP and the CNP
ieee8021QBridgeVlanStaticUntaggedPorts	- PortList containing just the VIP and the CNP

This sets the VIP and the CNP to be the only Ports present on the new I-component PVID. Furthermore, they are both set for untagged traffic. Thus, traffic arriving at one Port will be relayed to the other Port without any S-Tags.

The final step is to configure the ingress filtering on both the VIP and CNP. This is done by modifying their entries in the ieee8021QBridgePortVlanTable. For both of these Ports, the following columns in the table must be set:

ieee8021QBridgePortAcceptableFrameTypes	- acceptUntaggedAndPriority
ieee8021QBridgePortIngressFiltering	- true
ieee8021QBridgePVid	- IcompPVid

17.6.2.5 Service dependent provisioning for an S-tagged service interfaces

This subclause describes the MIB operations specific to S-tagged service interface provisioning. Configuration is done in a manner similar to Port-based services. The configuration differences are largely confined to different configurations of the CNP and VIP Port member sets, tagging, and ingress filtering.

The parameters for an S-tagged service interface are similar to those for a Port-based service interface.

17.6.2.5.1 Configuring the B-component

If the service interface to be created is unbundled, or if the service interface is bundled and the newly configured service interface is the first S-VID value to be transported on this Backbone I-SID, then the B-component will need to be configured.

The first step is to find the B-component ID and CBP for the traffic. Given the PIP, one uses the I-LAN table to find the B-component and CBP that the traffic for this service interface will transit. This gives us the following two parameters:

- BComponentId—B-component ID for the CBP
- CBPPortNumber—Port number of the CBP on this B-component

These, together with the BackboneSid, DefaultDestAddress, and BVid value can be used to configure the B-component as per the previous instructions. As a side effect of configuring the CBP via the ServiceMapping table, the ieee8021PbbVipDefaultDstBMAC column is set in the ieee8021PbbVipTable.

17.6.2.5.2 Configuring the I-component for S-tagged service interfaces

These service interfaces are configured by creating one or more static VLANs on the I-component whose S-VIDs are the values specified as the S-VID parameters to the service. For a one-to-one service, exactly one S-VID value is specified. For a many-to-one or all-to-one service, multiple S-VID values are specified.

This is done by creating an entry in the ieee8021QBridgeVlanStaticTable with the columns set to the following values:

ieee8021QBridgeVlanStaticComponentId	- IcomponentId
ieee8021QBridgeVlanStaticVlanIndex	- S-VID
ieee8021QBridgeVlanStaticName	- Something administratively convenient
ieee8021QBridgeVlanStaticEgressPorts	- Port List containing just the VIP and the CNP

For the one-to-one service set:

ieee8021QBridgeVlanStaticUntaggedPorts	- PortList containing just the VIP
--	------------------------------------

For the many-to-one or all-to-one service:

ieee8021QBridgeVlanStaticUntaggedPorts	- Null set
--	------------

For the unbundled one-to-one service, the VIP and CNP are set to be the only Ports that are members of the I-component's VLAN identified by S-VID. Adding the VIP to the set of untagged Ports means that the data will be sent to the CBP on the B-component untagged.

For the bundled many-to-one or all-to-one service interfaces, the VIP may be in the member set of more than one S-VID. This is the mechanism by which multiple CNP services, identified by S-VID, are mapped to the same I-SID value. Furthermore, unlike the unbundled service, the VIP is not a member of the untagged egress set of the S-VID.

17.7 MIB modules

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 MIB modules definitions below, if any discrepancy between the DESCRIPTION text and the corresponding definition in any other part of this standard occur, the definitions outside this subclause take precedence.

17.7.1 Definitions for the IEEE8021-TC MIB module

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

-- =====
-- TEXTUAL-CONVENTIONS MIB for IEEE 802.1
-- =====

IMPORTS
    MODULE-IDENTITY, Unsigned32, org
        FROM SNMPv2-SMI -- RFC 2578
    TEXTUAL-CONVENTION
        FROM SNMPv2-TC; -- RFC 2579

ieee8021TcMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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 Levi
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG

    Contact: Kevin Nolish
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
    "Textual conventions used throughout the various IEEE 802.1 MIB
modules.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
```

REVISION "201102270000Z" -- February 27, 2011
DESCRIPTION
"Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."

REVISION "200811180000Z" -- November 18, 2008
DESCRIPTION
"Added textual conventions needed to support the IEEE 802.1
MIBs for PBB-TE. Additionally, some textual conventions were
modified for the same reason."

REVISION "200810150000Z" -- October 15, 2008
DESCRIPTION
"Initial version."
 ::= { org ieee(111) standards-association-numbers-series-standards(2)
 lan-man-stds(802) ieee802dot1(1) 1 1 }

ieee802dot1mibs OBJECT IDENTIFIER
 ::= { org ieee(111) standards-association-numbers-series-standards(2)
 lan-man-stds(802) ieee802dot1(1) 1 }

-- =====
-- Textual Conventions
-- =====

IEEE8021PbbComponentIdentifier ::= TEXTUAL-CONVENTION
 DISPLAY-HINT "d"
 STATUS current
 DESCRIPTION
 "The component identifier is used to distinguish between the
 multiple virtual bridge instances within a PB or PBB. Each
 virtual bridge instance is called a component. In simple
 situations where there is only a single component the default
 value is 1. The component is identified by a component
 identifier unique within the BEB and by a MAC address unique
 within the PBBN. Each component is associated with a Backbone
 Edge Bridge (BEB) Configuration managed object."
 REFERENCE "12.3 1)"
 SYNTAX Unsigned32 (1..4294967295)

IEEE8021PbbComponentIdentifierOrZero ::= TEXTUAL-CONVENTION
 DISPLAY-HINT "d"
 STATUS current
 DESCRIPTION
 "The component identifier is used to distinguish between the
 multiple virtual bridge instances within a PB or PBB. In simple
 situations where there is only a single component the default
 value is 1. The component is identified by a component
 identifier unique within the BEB and by a MAC address unique
 within the PBBN. Each component is associated with a Backbone
 Edge Bridge (BEB) Configuration managed object.

The special value '0' means 'no component identifier'. When
 this TC is used as the SYNTAX of an object, that object must
 specify the exact meaning for this value."
 REFERENCE "12.3 1)"
 SYNTAX Unsigned32 (0 | 1..4294967295)

IEEE8021PbbServiceIdentifier ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"
STATUS current
DESCRIPTION
 "The service instance identifier is used at the Customer Backbone Port of a PBB to distinguish a service instance (Local-SID). If the Local-SID field is supported, it is used to perform a bidirectional 1:1 mapping between the Backbone I-SID and the Local-SID. If the Local-SID field is not supported, the Local-SID value is the same as the Backbone I-SID value."
REFERENCE "12.16.3, 12.16.5"
SYNTAX Unsigned32 (256..16777214)

IEEE8021PbbServiceIdentifierOrUnassigned ::= TEXTUAL-CONVENTION
DISPLAY-HINT "d"
STATUS current
DESCRIPTION
 "The service instance identifier is used at the Customer Backbone Port of a PBB to distinguish a service instance (Local-SID). If the Local-SID field is supported, it is used to perform a bidirectional 1:1 mapping between the Backbone I-SID and the Local-SID. If the Local-SID field is not supported, the Local-SID value is the same as the Backbone I-SID value.

The special value of 1 indicates an unassigned I-SID."
REFERENCE "12.16.3, 12.16.5"
SYNTAX Unsigned32 (1|256..16777214)

IEEE8021PbbIngressEgress ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
 "A 2 bit selector which determines if frames on this VIP may ingress to the PBBN but not egress the PBBN, egress to the PBBN but not ingress the PBBN, or both ingress and egress the PBBN."
REFERENCE "12.16.3, 12.16.5, 12.16.6"
SYNTAX BITS {
 ingress(0),
 egress(1)
}

IEEE8021PriorityCodePoint ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
 "Bridge ports may encode or decode the PCP value of the frames that traverse the port. This textual convention names the possible encoding and decoding schemes that the port may use. The priority and drop_eligible parameters are encoded in the Priority Code Point (PCP) field of the VLAN tag using the Priority Code Point Encoding Table for the Port, and they are decoded from the PCP using the Priority Code Point Decoding Table."
REFERENCE "12.6.2.6"
SYNTAX INTEGER {
 codePoint8p0d(1),
 codePoint7p1d(2),
 codePoint6p2d(3),
 codePoint5p3d(4)

}

```

IEEE8021BridgePortNumber ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "An integer that uniquely identifies a bridge port, as
         specified in 17.3.2.2 of IEEE 802.1ap.
         This value is used within the spanning tree
         protocol to identify this port to neighbor bridges."
    REFERENCE "17.3.2.2"
    SYNTAX      Unsigned32 (1..65535)

IEEE8021BridgePortNumberOrZero ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "An integer that uniquely identifies a bridge port, as
         specified in 17.3.2.2 of IEEE 802.1ap. The value 0
         means no port number, and this must be clarified in the
         DESCRIPTION clause of any object defined using this
         TEXTUAL-CONVENTION."
    REFERENCE "17.3.2.2"
    SYNTAX      Unsigned32 (0..65535)

IEEE8021BridgePortType ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "A port type. The possible port types are:

            customerVlanPort(2) - Indicates a port is a C-tag
                aware port of an enterprise VLAN aware bridge.

            providerNetworkPort(3) - Indicates a port is an S-tag
                aware port of a Provider Bridge or Backbone Edge
                Bridge used for connections within a PBN or PBBN.

            customerNetworkPort(4) - Indicates a port is an S-tag
                aware port of a Provider Bridge or Backbone Edge
                Bridge used for connections to the exterior of a
                PBN or PBBN.

            customerEdgePort(5) - Indicates a port is a C-tag
                aware port of a Provider Bridge used for connections
                to the exterior of a PBN or PBBN.

            customerBackbonePort(6) - Indicates a port is a I-tag
                aware port of a Backbone Edge Bridge's B-component.

            virtualInstancePort(7) - Indicates a port is a virtual
                S-tag aware port within a Backbone Edge Bridge's
                I-component which is responsible for handling
                S-tagged traffic for a specific backbone service
                instance.

            dBridgePort(8) - Indicates a port is a VLAN-unaware
                member of an 802.1D bridge."
    REFERENCE "12.16.1.1.3 h4), 12.16.2.1/2,
               12.13.1.1, 12.13.1.2, 12.15.2.1, 12.15.2.2"

```

```
SYNTAX      INTEGER {
            none(1),
            customerVlanPort(2),
            providerNetworkPort(3),
            customerNetworkPort(4),
            customerEdgePort(5),
            customerBackbonePort(6),
            virtualInstancePort(7),
            dBridgePort(8)
        }

IEEE8021VlanIndex ::= TEXTUAL-CONVENTION
DISPLAY-HINT "d"
STATUS      current
DESCRIPTION
    "A value used to index per-VLAN tables: values of 0 and
     4095 are not permitted. If the value is between 1 and
     4094 inclusive, it represents an IEEE 802.1Q VLAN-ID with
     global scope within a given bridged domain (see VlanId
     textual convention). If the value is greater than 4095,
     then it represents a VLAN with scope local to the
     particular agent, i.e., one without a global VLAN-ID
     assigned to it. Such VLANs are outside the scope of
     IEEE 802.1Q, but it is convenient to be able to manage them
     in the same way using this MIB."
REFERENCE   "9.6"
SYNTAX      Unsigned32 (1..4094|4096..4294967295)

IEEE8021VlanIndexOrWildcard ::= TEXTUAL-CONVENTION
DISPLAY-HINT "d"
STATUS      current
DESCRIPTION
    "A value used to index per-VLAN tables. The value 0 is not
     permitted, while the value 4095 represents a 'wildcard'
     value. An object whose SYNTAX is IEEE8021VlanIndexOrWildcard
     must specify in its DESCRIPTION the specific meaning of the
     wildcard value. If the value is between 1 and
     4094 inclusive, it represents an IEEE 802.1Q VLAN-ID with
     global scope within a given bridged domain (see VlanId
     textual convention). If the value is greater than 4095,
     then it represents a VLAN with scope local to the
     particular agent, i.e., one without a global VLAN-ID
     assigned to it. Such VLANs are outside the scope of
     IEEE 802.1Q, but it is convenient to be able to manage them
     in the same way using this MIB."
REFERENCE   "9.6"
SYNTAX      Unsigned32 (1..4294967295)

IEEE8021MstIdentifier ::= TEXTUAL-CONVENTION
DISPLAY-HINT "d"
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, an MSTID, i.e., a value used to identify
     a spanning tree (or MST) instance. In the PBB-TE environment
     the value 4094 is used to identify VIDs managed by the PBB-TE
     procedures."
SYNTAX      Unsigned32 (1..4094)

IEEE8021ServiceSelectorType ::= TEXTUAL-CONVENTION
```

STATUS current
 DESCRIPTION "A value that represents a type (and thereby the format) of a IEEE8021ServiceSelectorValue. 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.
vlanId(1)	12-Bit identifier as described in IEEE802.1Q.
isid(2)	24-Bit identifier as described in IEEE802.1ah.
tesid(3)	32 Bit identifier as described below
ieeeReserved(xx)	Reserved for definition by IEEE 802.1 xx values can be [4..7].

To support future extensions, the IEEE8021ServiceSelectorType 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.

The tesid is used as a service selector for MAs that are present in bridges that implement PBB-TE functionality. A selector of this type is interpreted as a 32 bit unsigned value of type IEEE8021PbbTeTSidId. This type is used to index the Ieee8021PbbTeTeSidTable to find the ESPs which comprise the TE Service Instance named by this TE-SID value.

Implementations MUST ensure that IEEE8021ServiceSelectorType objects and any dependent objects (e.g., IEEE8021ServiceSelectorValue objects) are consistent. An inconsistentValue error MUST be generated if an attempt to change an IEEE8021ServiceSelectorType object would, for example, lead to an undefined IEEE8021ServiceSelectorValue value."

SYNTAX INTEGER {
 vlanId(1),
 isid(2),
 tesid(3)
 }

IEEE8021ServiceSelectorValueOrNone ::= TEXTUAL-CONVENTION
 DISPLAY-HINT "d"
 STATUS current
 DESCRIPTION "An integer that uniquely identifies a generic MAC service, or none. Examples of service selectors are a VLAN-ID (IEEE 802.1Q) and an I-SID (IEEE 802.1ah)."

An IEEE8021ServiceSelectorValueOrNone value is always interpreted within the context of an IEEE8021ServiceSelectorType value. Every usage of the IEEE8021ServiceSelectorValueOrNone textual convention is required to specify the IEEE8021ServiceSelectorType object that provides the context. It is suggested that the IEEE8021ServiceSelectorType object be logically registered before the object(s) that use the IEEE8021ServiceSelectorValueOrNone textual convention, if they appear in the same logical row.

The value of an IEEE8021ServiceSelectorValueOrNone object must always be consistent with the value of the associated IEEE8021ServiceSelectorType object. Attempts to set an IEEE8021ServiceSelectorValueOrNone object to a value inconsistent with the associated IEEE8021ServiceSelectorType must fail with an inconsistentValue error.

The special value of zero is used to indicate that no service selector is present or used. This can be used in any situation where an object or a table entry MUST either refer to a specific service, or not make a selection.

Note that a MIB object that is defined using this TEXTUAL-CONVENTION SHOULD clarify the meaning of 'no service' (i.e., the special value 0), as well as the maximum value (i.e., 4094, for a VLAN ID)."

SYNTAX Unsigned32 (0 | 1..4294967295)

```
IEEE8021ServiceSelectorValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "An integer that uniquely identifies a generic MAC service.
         Examples of service selectors are a VLAN-ID (IEEE 802.1Q)
         and an I-SID (IEEE 802.1ah)."
```

An IEEE8021ServiceSelectorValue value is always interpreted within the context of an IEEE8021ServiceSelectorType value. Every usage of the IEEE8021ServiceSelectorValue textual convention is required to specify the IEEE8021ServiceSelectorType object that provides the context. It is suggested that the IEEE8021ServiceSelectorType object be logically registered before the object(s) that use the IEEE8021ServiceSelectorValue textual convention, if they appear in the same logical row.

The value of an IEEE8021ServiceSelectorValue object must always be consistent with the value of the associated IEEE8021ServiceSelectorType object. Attempts to set an IEEE8021ServiceSelectorValue object to a value inconsistent with the associated IEEE8021ServiceSelectorType must fail with an inconsistentValue error.

Note that a MIB object that is defined using this TEXTUAL-CONVENTION SHOULD clarify the maximum value (i.e., 4094, for a VLAN ID)."

SYNTAX Unsigned32 (1..4294967295)

```
IEEE8021PortAcceptableFrameTypes ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "Acceptable frame types on a port."
    REFERENCE  "12.10.1.3, 12.13.3.3, 12.13.3.4"
    SYNTAX     INTEGER {
        admitAll(1),
        admitUntaggedAndPriority(2),
        admitTagged(3)
    }
```

```
IEEE8021PriorityValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS current
    DESCRIPTION
        "An 802.1Q user priority value."
    REFERENCE "12.13.3.3"
    SYNTAX Unsigned32 (0..7)

IEEE8021PbbTeProtectionGroupId ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS current
    DESCRIPTION
        "The PbbTeProtectionGroupId identifier is used to distinguish
         protection group instances present in the B Component of
         an IB-BEB."
    REFERENCE "12.19.2"
    SYNTAX Unsigned32 (1..429467295)

IEEE8021PbbTeEsp ::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
        "This textual convention is used to represent the logical
         components that comprise the 3-tuple that identifies an
         Ethernet Switched Path. The 3-tuple consists of a
         destination MAC address, a source MAC address and a VID.
         Bytes (1..6) of this textual convention contain the
         ESP-MAC-DA, bytes (7..12) contain the ESP-MAC-SA, and bytes
         (13..14) contain the ESP-VID."
    REFERENCE "802.1Qay 3.2"
    SYNTAX OCTET STRING (SIZE(14))

IEEE8021PbbTeTSidId ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS current
    DESCRIPTION
        "This textual convention is used to represent an identifier
         that refers to a TE Service Instance. Note that, internally
         a TE-SID is implementation dependent. This textual convention
         defines the external representation of TE-SID values."
    REFERENCE
        "802.1Qay 3.11"
    SYNTAX Unsigned32 (1..42947295)

IEEE8021PbbTeProtectionGroupConfigAdmin ::= TEXTUAL-CONVENTION
    STATUS current
    DESCRIPTION
        "This textual convention is used to represent administrative
         commands that can be issued to a protection group. The value
         noAdmin(1) is used to indicate that no administrative action
         is to be performed."
    REFERENCE "26.10.3.3.5
                26.10.3.3.6
                26.10.3.3.7
                12.19.2.3.2"
    SYNTAX INTEGER {
        clear(1),
        lockOutProtection(2),
        forceSwitch(3),
```

```
        manualSwitchToProtection(4),
        manualSwitchToWorking(5)
    }

IEEE8021PbbTeProtectionGroupActiveRequests ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
    "This textual convention is used to represent the status of
     active requests within a protection group."
REFERENCE
    "12.19.2.1.3 d)"
SYNTAX  INTEGER {
        noRequest(1),
        loP(2),
        fs(3),
        pSFH(4),
        wSFH(5),
        manualSwitchToProtection(6),
        manualSwitchToWorking(7)
    }

END
```

17.7.2 Definitions for the IEEE8021-BRIDGE MIB module

```

IEEE8021-BRIDGE-MIB DEFINITIONS ::= BEGIN

-- =====
-- MIB for IEEE 802.1D devices
-- =====
IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    Integer32, Counter64
        FROM SNMPv2-SMI
    RowStatus, MacAddress, TruthValue, TimeInterval
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    ifIndex, InterfaceIndexOrZero, ifGeneralInformationGroup
        FROM IF-MIB
    ieee802dot1mibs, IEEE8021PbbComponentIdentifier,
    IEEE8021BridgePortNumber, IEEE8021PriorityCodePoint,
    IEEE8021BridgePortType, IEEE8021PriorityValue
        FROM IEEE8021-TC-MIB
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    systemGroup
        FROM SNMPv2-MIB
;

ieee8021BridgeMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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 Levi
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
"The Bridge MIB module for managing devices that support
IEEE 802.1D. This MIB module is derived from the IETF
BRIDGE-MIB, RFC 4188.

Unless otherwise indicated, the references in this MIB
module are to IEEE Std 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
REVISION      "201102270000Z" -- February 27, 2011
DESCRIPTION
    "Minor edits to contact information etc. as part of

```

2011 revision of IEEE Std 802.1Q."

REVISION "200810150000Z" -- October 15, 2008
DESCRIPTION
"Initial revision, derived from RFC 4188."
 ::= { ieee802dot1mibs 2 }

-- =====
-- subtrees in the Bridge MIB
-- =====

ieee8021BridgeNotifications
OBJECT IDENTIFIER ::= { ieee8021BridgeMib 0 }

ieee8021BridgeObjects
OBJECT IDENTIFIER ::= { ieee8021BridgeMib 1 }

ieee8021BridgeConformance
OBJECT IDENTIFIER ::= { ieee8021BridgeMib 2 }

ieee8021BridgeBase
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 1 }
ieee8021BridgeTp
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 2 }
ieee8021BridgePriority
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 3 }
ieee8021BridgeMrp
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 4 }
ieee8021BridgeMmrp
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 5 }
ieee8021BridgeInternalLan
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 6 }
ieee8021BridgeDot1d
OBJECT IDENTIFIER ::= { ieee8021BridgeObjects 7 }

-- =====
-- the ieee8021BridgeBase subtree
-- =====
-- Implementation of the ieee8021BridgeBase subtree is mandatory
-- for all bridges.
-- =====

-- =====
-- the ieee8021BridgeBaseTable
-- =====

ieee8021BridgeBaseTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021BridgeBaseEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"A table that contains generic information about every
bridge component. All writable objects in this table
must be persistent over power up restart/reboot."
REFERENCE "12.4.1"
 ::= { ieee8021BridgeBase 1 }

ieee8021BridgeBaseEntry OBJECT-TYPE
SYNTAX Ieee8021BridgeBaseEntry
MAX-ACCESS not-accessible

```

STATUS      current
DESCRIPTION
    "A list of objects containing information for each bridge
     component."
INDEX  { ieee8021BridgeBaseComponentId }
 ::= { ieee8021BridgeBaseTable 1 }

Ieee8021BridgeBaseEntry ::=
SEQUENCE {
    ieee8021BridgeBaseComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021BridgeBaseBridgeAddress
        MacAddress,
    ieee8021BridgeBaseNumPorts
        Integer32,
    ieee8021BridgeBaseComponentType
        INTEGER,
    ieee8021BridgeBaseDeviceCapabilities
        BITS,
    ieee8021BridgeBaseTrafficClassesEnabled
        TruthValue,
    ieee8021BridgeBaseMmrpEnabledStatus
        TruthValue,
    ieee8021BridgeBaseRowStatus
        RowStatus
}
ieee8021BridgeBaseComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The component identifier is used to distinguish between the
     multiple virtual bridge instances within a PBB. In simple
     situations where there is only a single component the default
     value is 1."
 ::= { ieee8021BridgeBaseEntry 1 }

ieee8021BridgeBaseBridgeAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The MAC address used by this bridge when it must be
     referred to in a unique fashion. It is recommended
     that this be the numerically smallest MAC address of
     all ports that belong to this bridge. However, it is
     only required to be unique. When concatenated with
     ieee8021SpanningTreePriority, a unique BridgeIdentifier
     is formed, which is used in the Spanning Tree Protocol.

This object may not be modified while the corresponding
instance of ieee8021BridgeBaseRowStatus is active(1).

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE   "12.4.1.1.3 a)"
 ::= { ieee8021BridgeBaseEntry 2 }

```

```
ieee8021BridgeBaseNumPorts OBJECT-TYPE
    SYNTAX      Integer32
    UNITS      "ports"
    MAX-ACCESS  read-only
    STATUS     current
    DESCRIPTION
        "The number of ports controlled by this bridging
         entity."
    REFERENCE   "12.4.1.1.3 c)"
    ::= { ieee8021BridgeBaseEntry 3 }

ieee8021BridgeBaseComponentType OBJECT-TYPE
    SYNTAX      INTEGER {
        iComponent(1),
        bComponent(2),
        cVlanComponent(3),
        sVlanComponent(4),
        dBridgeComponent(5)
    }
    MAX-ACCESS  read-create
    STATUS     current
    DESCRIPTION
        "Indicates the component type(s) of this bridge. The
         following component types are possible:
         iComponent(1) - An S-VLAN component of a Backbone
                         Edge Bridge which performs encapsulation of customer
                         frames.
         bComponent(2) - An S-VLAN component of a Backbone
                         Edge Bridge which bundles backbone service instances
                         into B-VLANs.
         cVlanComponent(3) - A C-VLAN component of an
                           enterprise VLAN bridge or of a Provider Bridge used
                           to process C-tagged frames.
         sVlanComponent(4) - An S-VLAN component of a
                           Provider Bridge.
         dBridgeComponent(5) - A VLAN unaware component of an
                           802.1D bridge.

This object may not be modified while the corresponding
instance of ieee8021BridgeBaseRowStatus is active(1).

The value of this object MUST be retained across
reinitializations of the management system."
    REFERENCE   "12.3 m)"
    ::= { ieee8021BridgeBaseEntry 4 }

ieee8021BridgeBaseDeviceCapabilities OBJECT-TYPE
    SYNTAX      BITS {
        dot1dExtendedFilteringServices(0),
        dot1dTrafficClasses(1),
        dot1qStaticEntryIndividualPort(2),
        dot1qIVLCapable(3),
        dot1qSVLCapable(4),
        dot1qHybridCapable(5),
```

```

dot1qConfigurablePvidTagging(6),
dot1dLocalVlanCapable(7)
}
MAX-ACCESS  read-create
STATUS       current
DESCRIPTION
  "Indicates the optional parts of IEEE 802.1D and 802.1Q
   that are implemented by this device and are manageable
   through this MIB. Capabilities that are allowed on a
   per-port basis are indicated in
   ieee8021BridgeBasePortCapabilities.

dot1dExtendedFilteringServices(0),
  -- can perform filtering of
  -- individual multicast addresses
  -- controlled by MMRP.
dot1dTrafficClasses(1),
  -- can map user priority to
  -- multiple traffic classes.
dot1qStaticEntryIndividualPort(2),
  -- dot1qStaticUnicastReceivePort &
  -- dot1qStaticMulticastReceivePort
  -- can represent non-zero entries.
dot1qIVLCapable(3),  -- Independent VLAN Learning (IVL).
dot1qSVLCapable(4), -- Shared VLAN Learning (SVL).
dot1qHybridCapable(5),
  -- both IVL & SVL simultaneously.
dot1qConfigurablePvidTagging(6),
  -- whether the implementation
  -- supports the ability to
  -- override the default PVID
  -- setting and its egress status
  -- (VLAN-Tagged or Untagged) on
  -- each port.
dot1dLocalVlanCapable(7)
  -- can support multiple local
  -- bridges, outside of the scope
  -- of 802.1Q defined VLANs.

```

This object may not be modified while the corresponding instance of ieee8021BridgeBaseRowStatus is active(1).

The value of this object MUST be retained across reinitializations of the management system."
 REFERENCE "12.10.1.1.3 b)"
 ::= { ieee8021BridgeBaseEntry 5 }

```

ieee8021BridgeBaseTrafficClassesEnabled OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "The value true(1) indicates that Traffic Classes are
   enabled on this bridge. When false(2), the bridge
   operates with a single priority level for all traffic.

```

This object may be modified while the corresponding instance of ieee8021BridgeBaseRowStatus is active(1).

The value of this object MUST be retained across reinitializations of the management system."

DEFVAL { true }
 ::= { ieee8021BridgeBaseEntry 6 }

ieee8021BridgeBaseMmrpEnabledStatus OBJECT-TYPE

SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION

"The administrative status requested by management for MMRP. The value true(1) indicates that MMRP should be enabled on this device, in all VLANs, on all ports for which it has not been specifically disabled. When false(2), MMRP is disabled, in all VLANs and on all ports, and all MMRP packets will be forwarded transparently. This object affects both Applicant and Registrar state machines. A transition from false(2) to true(1) will cause a reset of all MMRP state machines on all ports.

This object may be modified while the corresponding instance of ieee8021BridgeBaseRowStatus is active(1).

The value of this object MUST be retained across reinitializations of the management system."

DEFVAL { true }
 ::= { ieee8021BridgeBaseEntry 7 }

ieee8021BridgeBaseRowStatus OBJECT-TYPE

SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION

"The object indicates the status of an entry, and is used to create/delete entries.

The following objects must be set prior to making a new entry active:

 ieee8021BridgeBaseBridgeAddress
 ieee8021BridgeBaseComponentType
 ieee8021BridgeBaseDeviceCapabilities

It is recommended that these three objects not be allowed to be modified while the corresponding instance of ieee8021BridgeBaseRowStatus object is active(1).

The following objects are not required to be set before making a new entry active (they will take their defaults), and they also may be modified while the corresponding instance of this object is active(1):

 ieee8021BridgeBaseTrafficClassesEnabled
 ieee8021BridgeBaseMmrpEnabledStatus

The value of this object and all corresponding instances of other objects in this table MUST be retained across reinitializations of the management system."

::= { ieee8021BridgeBaseEntry 8 }

-- =====

```
-- The Generic Bridge Port Table
-- =====
ieee8021BridgeBasePortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021BridgeBasePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains generic information about every
         port that is associated with this bridge. Transparent,
         and source-route ports are included."
    REFERENCE   "12.4.2"
    ::= { ieee8021BridgeBase 4 }

ieee8021BridgeBasePortEntry OBJECT-TYPE
    SYNTAX      Ieee8021BridgeBasePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current

    DESCRIPTION
        "A list of objects containing information for each port
         of the bridge."
    INDEX   { ieee8021BridgeBasePortComponentId,
              ieee8021BridgeBasePort }
    ::= { ieee8021BridgeBasePortTable 1 }

Ieee8021BridgeBasePortEntry ::=
SEQUENCE {
    ieee8021BridgeBasePortComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021BridgeBasePort
        IEEE8021BridgePortNumber,
    ieee8021BridgeBasePortIfIndex
        InterfaceIndexOrZero,
    ieee8021BridgeBasePortDelayExceededDiscards
        Counter64,
    ieee8021BridgeBasePortMtuExceededDiscards
        Counter64,
    ieee8021BridgeBasePortCapabilities
        BITS,
    ieee8021BridgeBasePortTypeCapabilities
        BITS,
    ieee8021BridgeBasePortType
        IEEE8021BridgePortType,
    ieee8021BridgeBasePortExternal
        TruthValue,
    ieee8021BridgeBasePortAdminPointToPoint
        INTEGER,
    ieee8021BridgeBasePortOperPointToPoint
        TruthValue,
    ieee8021BridgeBasePortName
        SnmpAdminString
}
}

ieee8021BridgeBasePortComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
```

multiple virtual bridge instances within a PBB. In simple situations where there is only a single component the default value is 1."

::= { ieee8021BridgeBasePortEntry 1 }

ieee8021BridgeBasePort OBJECT-TYPE
SYNTAX IEEE8021BridgePortNumber
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The port number of the port for which this entry contains bridge management information."
REFERENCE "12.4.2.1.2 a)"
 ::= { ieee8021BridgeBasePortEntry 2 }

ieee8021BridgeBasePortIfIndex OBJECT-TYPE
SYNTAX InterfaceIndexOrZero
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The value of the instance of the IfIndex object, defined in the IF-MIB, for the interface corresponding to this port, or the value 0 if the port has not been bound to an underlying frame source and sink.

It is an implementation specific decision as to whether this object may be modified if it has been created or if 0 is a legal value.

The underlying IfEntry indexed by this column must be persistent across reinitializations of the management system."
 ::= { ieee8021BridgeBasePortEntry 3 }

ieee8021BridgeBasePortDelayExceededDiscards OBJECT-TYPE
SYNTAX Counter64
UNITS "frames"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of frames discarded by this port due to excessive transit delay through the bridge. It is incremented by both transparent and source route bridges.

Discontinuities in the value of the counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any)."
REFERENCE "12.6.1.1.3 f)"
 ::= { ieee8021BridgeBasePortEntry 4 }

ieee8021BridgeBasePortMtuExceededDiscards OBJECT-TYPE
SYNTAX Counter64
UNITS "frames"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of frames discarded by this port due to an excessive size. It is incremented by both

transparent and source route bridges.

Discontinuities in the value of the counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any)."

REFERENCE "12.6.1.1.3 g)"
 ::= { ieee8021BridgeBasePortEntry 5 }

```
ieee8021BridgeBasePortCapabilities OBJECT-TYPE
SYNTAX      BITS {
    dot1qDot1qTagging(0),
    dot1qConfigurableAcceptableFrameTypes(1),
    dot1qIngressFiltering(2)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "Indicates the parts of IEEE 802.1D and 802.1Q that are optional on a per-port basis, that are implemented by this device, and that are manageable through this MIB.

  dot1qDot1qTagging(0), -- supports 802.1Q VLAN tagging of
                        -- frames and MVRP.
  dot1qConfigurableAcceptableFrameTypes(1),
                        -- allows modified values of
                        -- dot1qPortAcceptableFrameTypes.
  dot1qIngressFiltering(2)
                        -- supports the discarding of any
                        -- frame received on a Port whose
                        -- VLAN classification does not
                        -- include that Port in its Member
                        -- set."
REFERENCE  "12.10.1.1.3 c)"
 ::= { ieee8021BridgeBasePortEntry 6 }
```

```
ieee8021BridgeBasePortTypeCapabilities OBJECT-TYPE
SYNTAX      BITS {
    customerVlanPort(0),
    providerNetworkPort(1),
    customerNetworkPort(2),
    customerEdgePort(3),
    customerBackbonePort(4),
    virtualInstancePort(5),
    dBridgePort(6)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "Indicates the capabilities of this port. The corresponding instance of ieee8021BridgeBasePortType can potentially take any of the values for which the corresponding bit in this object is 1. The possible port types are as follows:

  customerVlanPort(0) - Indicates the port can be a C-tag aware port of an enterprise VLAN aware bridge.

  providerNetworkPort(1) - Indicates the port can be an
```

S-tag aware port of a Provider Bridge or Backbone Edge Bridge used for connections within a PBN or PBBN.

customerNetworkPort(2) - Indicates the port can be an S-tag aware port of a Provider Bridge or Backbone Edge Bridge used for connections to the exterior of a PBN or PBBN.

customerEdgePort(3) - Indicates the port can be a C-tag aware port of a Provider Bridge used for connections to the exterior of a PBN or PBBN.

customerBackbonePort(4) - Indicates the port can be a I-tag aware port of a Backbone Edge Bridge's B-component.

virtualInstancePort(5) - Indicates the port can be a virtual S-tag aware port within a Backbone Edge Bridge's I-component which is responsible for handling S-tagged traffic for a specific backbone service instance.

dBridgePort(6) - Indicates the port can be a VLAN-unaware member of an 802.1D bridge."

REFERENCE "12.16.1.1.3 h4), 12.16.2.1/2,
12.13.1.1, 12.13.1.2, 12.15.2.1, 12.15.2.2"
 ::= { ieee8021BridgeBasePortEntry 7 }

ieee8021BridgeBasePortType OBJECT-TYPE
SYNTAX IEEE8021BridgePortType
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The port type. This value must be persistent over power up restart/reboot."
REFERENCE "12.16.1.1.3 h4), 12.16.2.1/2,
12.13.1.1, 12.13.1.2, 12.15.2.1, 12.15.2.2"
 ::= { ieee8021BridgeBasePortEntry 8 }

ieee8021BridgeBasePortExternal OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION "A boolean indicating whether the port is external. A value of true(1) means the port is external. A value of false(2) means the port is internal."
REFERENCE "12.16.1.1.3 h4)"
 ::= { ieee8021BridgeBasePortEntry 9 }

ieee8021BridgeBasePortAdminPointToPoint OBJECT-TYPE
SYNTAX INTEGER {
forceTrue(1),
forceFalse(2),
auto(3)
}
MAX-ACCESS read-write
STATUS current

DESCRIPTION

"For a port running spanning tree, this object represents the administrative point-to-point status of the LAN segment attached to this port, using the enumeration values of 6.4.3. A value of forceTrue(1) indicates that this port should always be treated as if it is connected to a point-to-point link. A value of forceFalse(2) indicates that this port should be treated as having a shared media connection. A value of auto(3) indicates that this port is considered to have a point-to-point link if it is an Aggregator and all of its members are aggregatable, or if the MAC entity is configured for full duplex operation, either through auto-negotiation or by management means. Manipulating this object changes the underlying adminPointToPointMAC.

For a VIP, the adminPointToPointMAC parameter controls the mechanism by which the Default Backbone Destination parameter for the VIP is determined. For a backbone service instance that includes only 2 VIPs, the value may be set to forceTrue(1) which permits dynamic learning of the Default Backbone Destination parameter. For a backbone service instance that includes more than 2 VIPs, the value must be set to ForceFalse(2) or auto(3).

When this object is set to forceTrue(1) for a VIP, the Default Backbone Destination parameter is modified by the subsequent M_UNITDATA.indications as specified in 6.10.1 (and described in 26.4.1). Whenever the parameter is set to ForceFalse(2) or auto(3), the value for the Default Backbone Destination parameter is set to the Backbone Service Instance Group Address for the VIP-ISID.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "6.6.3, 6.10, 12.8.2.1.3 o), 12.8.2.3.2 f), 26.4.1"
 DEFVAL { forceFalse }
 ::= { ieee8021BridgeBasePortEntry 10 }

ieee8021BridgeBasePortOperPointToPoint OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"For a port running spanning tree, this object represents the operational point-to-point status of the LAN segment attached to this port. It indicates whether a port is considered to have a point-to-point connection. If adminPointToPointMAC is set to auto(2), then the value of operPointToPointMAC is determined in accordance with the specific procedures defined for the MAC entity concerned, as defined in 6.5 of IEEE 802.1w. The value is determined dynamically; that is, it is re-evaluated whenever the value of adminPointToPointMAC changes, and whenever the specific procedures defined for the MAC entity evaluate a change in its point-to-point status.

For a VIP, this object simply reflects the value of the corresponding instance of ieee8021BridgeBasePortAdminPointToPoint.

```
The value will be true(1) if that object is forceTrue(1), and
the value will be false(2) if the value of that object is either
forceFalse(2) or auto(3)."
REFERENCE "6.6.3, 6.10, 12.8.2.1.3 p), 12.8.2.3.2 f), 26.4.1"
 ::= { ieee8021BridgeBasePortEntry 11 }

ieee8021BridgeBasePortName OBJECT-TYPE
    SYNTAX     SnmpAdminString
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "A text string of up to 32 characters, of locally determined
significance."
    REFERENCE "12.4.2.1.3 a)"
    ::= { ieee8021BridgeBasePortEntry 12 }

-- =====
-- the ieee8021BridgeTp subtree
-- =====
-- This is implemented by those bridges that support the
-- transparent bridging mode. A transparent bridge will
-- implement this subtree.
-- =====

-- =====
-- Port Table for Transparent Bridges
-- =====

ieee8021BridgeTpPortTable OBJECT-TYPE
    SYNTAX     SEQUENCE OF Ieee8021BridgeTpPortEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "A table that contains information about every port that
is associated with this transparent bridge."
    REFERENCE "12.4.2, C.4"
    ::= { ieee8021BridgeTp 1 }

Ieee8021BridgeTpPortEntry OBJECT-TYPE
    SYNTAX     Ieee8021BridgeTpPortEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "A list of objects containing information for each port of
a transparent bridge."
    INDEX     { ieee8021BridgeTpPortComponentId,
                ieee8021BridgeTpPort }
    ::= { ieee8021BridgeTpPortTable 1 }

Ieee8021BridgeTpPortEntry :=
    SEQUENCE {
        ieee8021BridgeTpPortComponentId
            IEEE8021PbbComponentIdentifier,
        ieee8021BridgeTpPort
            IEEE8021BridgePortNumber,
        ieee8021BridgeTpPortMaxInfo
            Integer32,
        ieee8021BridgeTpPortInFrames
            Counter64,
```

```

        ieee8021BridgeTpPortOutFrames
            Counter64,
        ieee8021BridgeTpPortInDiscards
            Counter64
    }

ieee8021BridgeTpPortComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
        multiple virtual bridge instances within a PBB. In simple
        situations where there is only a single component the default
        value is 1."
    ::= { ieee8021BridgeTpPortEntry 1 }

ieee8021BridgeTpPort OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The port number of the port for which this entry
        contains Transparent bridging management information."
    ::= { ieee8021BridgeTpPortEntry 2 }

ieee8021BridgeTpPortMaxInfo OBJECT-TYPE
    SYNTAX      Integer32
    UNITS      "bytes"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum size of the INFO (non-MAC) field that
        this port will receive or transmit."
    ::= { ieee8021BridgeTpPortEntry 3 }

ieee8021BridgeTpPortInFrames OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of frames that have been received by this
        port from its segment. Note that a frame received on the
        interface corresponding to this port is only counted by
        this object if and only if it is for a protocol being
        processed by the local bridging function, including
        bridge management frames.

        Discontinuities in the value of the counter can occur
        at re-initialization of the management system, and at
        other times as indicated by the value of
        ifCounterDiscontinuityTime object of the associated
        interface (if any)."
    REFERENCE   "12.6.1.1.3 a)"
    ::= { ieee8021BridgeTpPortEntry 4 }

ieee8021BridgeTpPortOutFrames OBJECT-TYPE
    SYNTAX      Counter64

```

```
UNITS      "frames"
MAX-ACCESS  read-only
STATUS     current
DESCRIPTION
    "The number of frames that have been transmitted by this
    port to its segment. Note that a frame transmitted on
    the interface corresponding to this port is only counted
    by this object if and only if it is for a protocol being
    processed by the local bridging function, including
    bridge management frames.

    Discontinuities in the value of the counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    ifCounterDiscontinuityTime object of the associated
    interface (if any)."
REFERENCE  "12.6.1.1.3 d)"
 ::= { ieee8021BridgeTpPortEntry 5 }

ieee8021BridgeTpPortInDiscards OBJECT-TYPE
SYNTAX     Counter64
UNITS      "frames"
MAX-ACCESS  read-only
STATUS     current
DESCRIPTION
    "Count of received valid frames that were discarded
    (i.e., filtered) by the Forwarding Process.

    Discontinuities in the value of the counter can occur
    at re-initialization of the management system, and at
    other times as indicated by the value of
    ifCounterDiscontinuityTime object of the associated
    interface (if any)."
REFERENCE  "12.6.1.1.3 c)"
 ::= { ieee8021BridgeTpPortEntry 6 }

-- =====
-- the ieee8021BridgePriority subtree
-- =====

-- =====
-- Port Priority Table
-- =====

ieee8021BridgePortPriorityTable OBJECT-TYPE
SYNTAX     SEQUENCE OF Ieee8021BridgePortPriorityEntry
MAX-ACCESS  not-accessible
STATUS     current
DESCRIPTION
    "A table that contains information about every port that
    is associated with this transparent bridge."
 ::= { ieee8021BridgePriority 1 }

ieee8021BridgePortPriorityEntry OBJECT-TYPE
SYNTAX     Ieee8021BridgePortPriorityEntry
MAX-ACCESS  not-accessible
STATUS     current
DESCRIPTION
    "A list of Default User Priorities for each port of a
```

```

transparent bridge. This is indexed by
ieee8021BridgeBasePortComponentId and
ieee8021BridgeBasePort."
AUGMENTS { ieee8021BridgeBasePortEntry }
 ::= { ieee8021BridgePortPriorityTable 1 }

Ieee8021BridgePortPriorityEntry ::==
SEQUENCE {
    ieee8021BridgePortDefaultUserPriority
        IEEE8021PriorityValue,
    ieee8021BridgePortNumTrafficClasses
        Integer32,
    ieee8021BridgePortPriorityCodePointSelection
        IEEE8021PriorityCodePoint,
    ieee8021BridgePortUseDEI
        TruthValue,
    ieee8021BridgePortRequireDropEncoding
        TruthValue,
    ieee8021BridgePortServiceAccessPrioritySelection
        TruthValue
}

ieee8021BridgePortDefaultUserPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"The default ingress User Priority for this port. This
only has effect on media, such as Ethernet, that do not
support native User Priority.

The value of this object MUST be retained across
reinitializations of the management system."
 ::= { ieee8021BridgePortPriorityEntry 1 }

ieee8021BridgePortNumTrafficClasses OBJECT-TYPE
SYNTAX      Integer32 (1..8)
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"The number of egress traffic classes supported on this
port. This object may optionally be read-only.

The value of this object MUST be retained across
reinitializations of the management system."
 ::= { ieee8021BridgePortPriorityEntry 2 }

ieee8021BridgePortPriorityCodePointSelection OBJECT-TYPE
SYNTAX      IEEE8021PriorityCodePoint
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"This object identifies the rows in the PCP encoding and
decoding tables that are used to remark frames on this
port if this remarking is enabled."
REFERENCE   "12.6.2.6, 12.6.2.7"
 ::= { ieee8021BridgePortPriorityEntry 3 }

ieee8021BridgePortUseDEI OBJECT-TYPE

```

```
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "If the Use_DEI is set to true(1) for the Port then the
    drop_eligible parameter is encoded in the DEI of transmitted
    frames, and the drop_eligible parameter shall be true(1) for a
    received frame if the DEI is set in the VLAN tag or the Priority
    Code Point Decoding Table indicates drop_eligible True for
    the received PCP value. If the Use_DEI parameter is false(2),
    the DEI shall be transmitted as zero and ignored on receipt.
    The default value of the Use_DEI parameter is false(2)."
REFERENCE   "12.6.2.12, 12.6.2.13"
 ::= { ieee8021BridgePortPriorityEntry 4 }
```

```
ieee8021BridgePortRequireDropEncoding OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "If a Bridge supports encoding or decoding of drop_eligible
    from the PCP field of a VLAN tag (6.7.3) on any of its Ports,
    then it shall implement a Boolean parameter Require Drop
    Encoding on each of its Ports with default value false(2). If
    Require Drop Encoding is True and the Bridge Port cannot
    encode particular priorities with drop_eligible, then frames
    queued with those priorities and drop_eligible true(1) shall
    be discarded and not transmitted."
REFERENCE   "12.6.2.14, 12.6.2.15"
DEFVAL { false }
 ::= { ieee8021BridgePortPriorityEntry 5 }
```

```
ieee8021BridgePortServiceAccessPrioritySelection OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "Indication of if the Service Access Priority Selection
    function is supported on the Customer Bridge Port to request
    priority handling of the frame from a Port-based service
    interface."
REFERENCE   "12.6.2.16, 12.6.2.17"
 ::= { ieee8021BridgePortPriorityEntry 6 }
```

```
-- =====
-- User Priority Regeneration Table
-- =====
```

```
ieee8021BridgeUserPriorityRegenTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021BridgeUserPriorityRegenEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A list of Regenerated User Priorities for each received
    User Priority on each port of a bridge. The Regenerated
    User Priority value may be used to index the Traffic
    Class Table for each input port. This only has effect
    on media that support native User Priority. The default
    values for Regenerated User Priorities are the same as
```

```
        the User Priorities."
REFERENCE    "6.5"
 ::= { ieee8021BridgePriority 2 }

ieee8021BridgeUserPriorityRegenEntry OBJECT-TYPE
SYNTAX      Ieee8021BridgeUserPriorityRegenEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A mapping of incoming User Priority to a Regenerated
     User Priority."
INDEX      { ieee8021BridgeBasePortComponentId,
              ieee8021BridgeBasePort,
              ieee8021BridgeUserPriority }
 ::= { ieee8021BridgeUserPriorityRegenTable 1 }

Ieee8021BridgeUserPriorityRegenEntry :=
SEQUENCE {
    ieee8021BridgeUserPriority
        IEEE8021PriorityValue,
    ieee8021BridgeRegenUserPriority
        IEEE8021PriorityValue
}
 ::= { ieee8021BridgeUserPriorityRegenEntry 1 }

ieee8021BridgeUserPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The User Priority for a frame received on this port."
 ::= { ieee8021BridgeUserPriorityRegenEntry 1 }

ieee8021BridgeRegenUserPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The Regenerated User Priority that the incoming User
     Priority is mapped to for this port.

    The value of this object MUST be retained across
     reinitializations of the management system."
 ::= { ieee8021BridgeUserPriorityRegenEntry 2 }

-- =====
-- Traffic Class Table
-- =====

ieee8021BridgeTrafficClassTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021BridgeTrafficClassEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table mapping evaluated User Priority to Traffic
     Class, for forwarding by the bridge. Traffic class is a
     number in the range (0..(ieee8021BridgePortNumTrafficClasses-1))."
REFERENCE    "Table 8-3"
 ::= { ieee8021BridgePriority 3 }
```

```
ieee8021BridgeTrafficClassEntry OBJECT-TYPE
    SYNTAX      Ieee8021BridgeTrafficClassEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "User Priority to Traffic Class mapping."
INDEX   { ieee8021BridgeBasePortComponentId,
          ieee8021BridgeBasePort,
          ieee8021BridgeTrafficClassPriority }
 ::= { ieee8021BridgeTrafficClassTable 1 }

Ieee8021BridgeTrafficClassEntry ::==
SEQUENCE {
    ieee8021BridgeTrafficClassPriority
        IEEE8021PriorityValue,
    ieee8021BridgeTrafficClass
        Integer32
}

ieee8021BridgeTrafficClassPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Priority value determined for the received frame.
        This value is equivalent to the priority indicated in
        the tagged frame received, or one of the evaluated
        priorities, determined according to the media-type.
        For untagged frames received from Ethernet media, this
        value is equal to the ieee8021BridgePortDefaultUserPriority value
        for the ingress port.

        For untagged frames received from non-Ethernet media,
        this value is equal to the ieee8021BridgeRegenUserPriority value
        for the ingress port and media-specific user priority."
 ::= { ieee8021BridgeTrafficClassEntry 1 }

ieee8021BridgeTrafficClass OBJECT-TYPE
    SYNTAX      Integer32 (0..7)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The Traffic Class the received frame is mapped to.

        The value of this object MUST be retained across
        reinitializations of the management system."
 ::= { ieee8021BridgeTrafficClassEntry 2 }

-- =====
-- Outbound Access Priority Table
-- =====

ieee8021BridgePortOutboundAccessPriorityTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021BridgePortOutboundAccessPriorityEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table mapping Regenerated User Priority to Outbound
        Access Priority. This is a fixed mapping for all port
```

```

types, with two options for 802.5 Token Ring, and three
options for 802.17 RPR."
REFERENCE "Table 8-3"
 ::= { ieee8021BridgePriority 4 }

ieee8021BridgePortOutboundAccessPriorityEntry OBJECT-TYPE
  SYNTAX     Ieee8021BridgePortOutboundAccessPriorityEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "Regenerated User Priority to Outbound Access Priority
     mapping."
  INDEX    { ieee8021BridgeBasePortComponentId,
             ieee8021BridgeBasePort,
             ieee8021BridgeRegenUserPriority }
 ::= { ieee8021BridgePortOutboundAccessPriorityTable 1 }

Ieee8021BridgePortOutboundAccessPriorityEntry :=
  SEQUENCE {
    ieee8021BridgePortOutboundAccessPriority
    IEEE8021PriorityValue
  }

ieee8021BridgePortOutboundAccessPriority OBJECT-TYPE
  SYNTAX     IEEE8021PriorityValue
  MAX-ACCESS read-only
  STATUS     current
  DESCRIPTION
    "The Outbound Access Priority the received frame is
     mapped to."
 ::= { ieee8021BridgePortOutboundAccessPriorityEntry 1 }

-- =====
-- ieee8021BridgePortDecodingTable:
-- =====

ieee8021BridgePortDecodingTable OBJECT-TYPE
  SYNTAX     SEQUENCE OF Ieee8021BridgePortDecodingEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "A table that contains information about Priority Code
     Point Decoding Table for a Port of a provider bridge.
     Alternative values for each table are specified as rows
     in Table 6-4 (6.7.3), with each alternative labeled by
     the number of distinct priorities that can be communicated,
     and the number of these for which drop precedence can
     be communicated. All writable objects in this table must
     be persistent over power up restart/reboot."
 ::= { ieee8021BridgePriority 5 }

ieee8021BridgePortDecodingEntry OBJECT-TYPE
  SYNTAX     Ieee8021BridgePortDecodingEntry
  MAX-ACCESS not-accessible
  STATUS     current
  DESCRIPTION
    "A list of objects containing Priority Code Point Decoding
     information for a port of a provider bridge."
  INDEX { ieee8021BridgePortDecodingComponentId,

```

```
        ieee8021BridgePortDecodingPortNum,
        ieee8021BridgePortDecodingPriorityCodePointRow,
        ieee8021BridgePortDecodingPriorityCodePoint }
 ::= { ieee8021BridgePortDecodingTable 1 }

Ieee8021BridgePortDecodingEntry ::= SEQUENCE {
    ieee8021BridgePortDecodingComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021BridgePortDecodingPortNum
        IEEE8021BridgePortNumber,
    ieee8021BridgePortDecodingPriorityCodePointRow
        IEEE8021PriorityCodePoint,
    ieee8021BridgePortDecodingPriorityCodePoint
        Integer32,
    ieee8021BridgePortDecodingPriority
        IEEE8021PriorityValue,
    ieee8021BridgePortDecodingDropEligible
        TruthValue
}

ieee8021BridgePortDecodingComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
 ::= { ieee8021BridgePortDecodingEntry 1 }

ieee8021BridgePortDecodingPortNum OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique identifier of a port controlled by this VLAN
         bridging entity."
 ::= { ieee8021BridgePortDecodingEntry 2 }

ieee8021BridgePortDecodingPriorityCodePointRow OBJECT-TYPE
    SYNTAX      IEEE8021PriorityCodePoint
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The specific row in Table 6-3 (6.7.3) indicating the PCP."
 ::= { ieee8021BridgePortDecodingEntry 3 }

ieee8021BridgePortDecodingPriorityCodePoint OBJECT-TYPE
    SYNTAX      Integer32 (0..7)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The specific PCP value in Table 6-3 (6.7.3)."
 ::= { ieee8021BridgePortDecodingEntry 4 }

ieee8021BridgePortDecodingPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  read-write
```

```

STATUS      current
DESCRIPTION
    "The specific priority value in Table 6-3 (6.7.3)."
REFERENCE   "12.6.2.8, 12.6.2.9"
 ::= { ieee8021BridgePortDecodingEntry 5 }

ieee8021BridgePortDecodingDropEligible OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The drop eligibility value in Table 6-3 (6.7.3)."
REFERENCE   "12.6.2.8, 12.6.2.9"
 ::= { ieee8021BridgePortDecodingEntry 6 }

-- =====
-- ieee8021BridgePortEncodingTable:
-- =====

ieee8021BridgePortEncodingTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021BridgePortEncodingEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table that contains information about Priority Code
    Point Decoding Table for a Port of a provider bridge.
    Alternative values for each table are specified as rows
    in Table 6-3 (6.7.3), with each alternative labeled by
    the number of distinct priorities that can be communicated,
    and the number of these for which drop precedence can be
    communicated. All writable objects in this table must be
    persistent over power up restart/reboot."
 ::= { ieee8021BridgePriority 6 }

ieee8021BridgePortEncodingEntry OBJECT-TYPE
SYNTAX      Ieee8021BridgePortEncodingEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A list of objects containing Priority Code Point Encoding
    information for a port of a provider bridge."
INDEX { ieee8021BridgePortEncodingComponentId,
        ieee8021BridgePortEncodingPortNum,
        ieee8021BridgePortEncodingPriorityCodePointRow,
        ieee8021BridgePortEncodingPriorityCodePoint,
        ieee8021BridgePortEncodingDropEligible }
 ::= { ieee8021BridgePortEncodingTable 1 }

Ieee8021BridgePortEncodingEntry ::= SEQUENCE {
    ieee8021BridgePortEncodingComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021BridgePortEncodingPortNum
        IEEE8021BridgePortNumber,
    ieee8021BridgePortEncodingPriorityCodePointRow
        IEEE8021PriorityCodePoint,
    ieee8021BridgePortEncodingPriorityCodePoint
        Integer32,
    ieee8021BridgePortEncodingDropEligible
        TruthValue,
}

```

```
ieee8021BridgePortEncodingPriority
    IEEE8021PriorityValue
}

ieee8021BridgePortEncodingComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021BridgePortEncodingEntry 1 }

ieee8021BridgePortEncodingPortNum OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique identifier of a port controlled by this VLAN bridging
         entity."
    ::= { ieee8021BridgePortEncodingEntry 2 }

ieee8021BridgePortEncodingPriorityCodePointRow OBJECT-TYPE
    SYNTAX      IEEE8021PriorityCodePoint
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The specific row in Table 6-3 (6.7.3) indicating the PCP row.
         (i.e. 8P0D, 7P1D, 6P2D, 5P3D)"
    ::= { ieee8021BridgePortEncodingEntry 3 }

ieee8021BridgePortEncodingPriorityCodePoint OBJECT-TYPE
    SYNTAX      Integer32 (0..7)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The specific row in Table 6-3 (6.7.3) indicating the PCP.
         (i.e., 0,1,2,3,4,5,6,7)."
    ::= { ieee8021BridgePortEncodingEntry 4 }

ieee8021BridgePortEncodingDropEligible OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The specific row in Table 6-3 (6.7.3) indicating the drop
         eligibility. A value of true(1) means eligible for drop."
    ::= { ieee8021BridgePortEncodingEntry 5 }

ieee8021BridgePortEncodingPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The encoding priority in Table 6-3 (6.7.3)."
    REFERENCE   "12.6.2.10, 12.6.2.11"
    ::= { ieee8021BridgePortEncodingEntry 6 }
```

```
-- =====
-- ieee8021BridgeServiceAccessPriorityTable:
-- =====

ieee8021BridgeServiceAccessPriorityTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021BridgeServiceAccessPriorityEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains information about the Service Access Priority Selection function for a provider bridge. The use of this table enables a mechanism for a Customer Bridge attached to a Provider Bridged Network to request priority handling of frames. All writable objects in this table must be persistent over power up/restart/reboot."
 ::= { ieee8021BridgePriority 7 }

ieee8021BridgeServiceAccessPriorityEntry OBJECT-TYPE
    SYNTAX      Ieee8021BridgeServiceAccessPriorityEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing information about the Service Access Priority Selection function for a provider bridge."
INDEX { ieee8021BridgeServiceAccessPriorityComponentId,
         ieee8021BridgeServiceAccessPriorityPortNum,
         ieee8021BridgeServiceAccessPriorityReceived }
 ::= { ieee8021BridgeServiceAccessPriorityTable 1 }

Ieee8021BridgeServiceAccessPriorityEntry ::= SEQUENCE {
    ieee8021BridgeServiceAccessPriorityComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021BridgeServiceAccessPriorityPortNum
        IEEE8021BridgePortNumber,
    ieee8021BridgeServiceAccessPriorityReceived
        IEEE8021PriorityValue,
    ieee8021BridgeServiceAccessPriorityValue
        IEEE8021PriorityValue
}
}

ieee8021BridgeServiceAccessPriorityComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the multiple virtual bridge instances within a PBB. In simple situations where there is only a single component the default value is 1."
 ::= { ieee8021BridgeServiceAccessPriorityEntry 1 }

ieee8021BridgeServiceAccessPriorityPortNum OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A unique identifier of a port controlled by this VLAN bridging entity."

```

```
 ::= { ieee8021BridgeServiceAccessPriorityEntry 2 }

ieee8021BridgeServiceAccessPriorityReceived OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The default received priority value in Table 6-3 (6.7.3).
         (i.e., 0,1,2,3,4,5,6,7)"
    ::= { ieee8021BridgeServiceAccessPriorityEntry 3 }

ieee8021BridgeServiceAccessPriorityValue OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The regenerated priority value in Table 6-3 (6.7.3).
         (i.e., 0,1,2,3,4,5,6,7)"
    REFERENCE   "12.6.2.18, 12.6.2.19"
    ::= { ieee8021BridgeServiceAccessPriorityEntry 4 }

-- =====
-- the ieee8021BridgeMrp subtree
-- =====

-- =====
-- The MRP Port Table
-- =====

ieee8021BridgePortMrpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021BridgePortMrpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table of MRP control information about every bridge
         port. This is indexed by ieee8021BridgeBasePortComponentId
         and ieee8021BridgeBasePort."
    ::= { ieee8021BridgeMrp 1 }

ieee8021BridgePortMrpEntry OBJECT-TYPE
    SYNTAX      Ieee8021BridgePortMrpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "MRP control information for a bridge port."
    AUGMENTS { ieee8021BridgeBasePortEntry }
    ::= { ieee8021BridgePortMrpTable 1 }

Ieee8021BridgePortMrpEntry ::=
    SEQUENCE {
        ieee8021BridgePortMrpJoinTime
            TimeInterval,
        ieee8021BridgePortMrpLeaveTime
            TimeInterval,
        ieee8021BridgePortMrpLeaveAllTime
            TimeInterval
    }

ieee8021BridgePortMrpJoinTime OBJECT-TYPE
```

```

SYNTAX      TimeInterval
UNITS       "centi-seconds"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The MRP Join time, in centiseconds.

    The value of this object MUST be retained across
    reinitializations of the management system."
DEFVAL      { 20 }
 ::= { ieee8021BridgePortMrpEntry 1 }

ieee8021BridgePortMrpLeaveTime OBJECT-TYPE
SYNTAX      TimeInterval
UNITS       "centi-seconds"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The MRP Leave time, in centiseconds.

    The value of this object MUST be retained across
    reinitializations of the management system."
DEFVAL      { 60 }
 ::= { ieee8021BridgePortMrpEntry 2 }

ieee8021BridgePortMrpLeaveAllTime OBJECT-TYPE
SYNTAX      TimeInterval
UNITS       "centi-seconds"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The MRP LeaveAll time, in centiseconds.

    The value of this object MUST be retained across
    reinitializations of the management system."
DEFVAL      { 1000 }
 ::= { ieee8021BridgePortMrpEntry 3 }

-- =====
-- The MMRP Port Configuration and Status Table
-- =====

ieee8021BridgePortMmrpTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021BridgePortMmrpEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table of MMRP control and status information about
    every bridge port.  Augments the ieee8021BridgeBasePortTable."
 ::= { ieee8021BridgeMmrp 1 }

ieee8021BridgePortMmrpEntry OBJECT-TYPE
SYNTAX      Ieee8021BridgePortMmrpEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "MMRP control and status information for a bridge port."
AUGMENTS { ieee8021BridgeBasePortEntry }
 ::= { ieee8021BridgePortMmrpTable 1 }

```

```
Ieee8021BridgePortMmrpEntry ::=  
SEQUENCE {  
    ieee8021BridgePortMmrpEnabledStatus  
        TruthValue,  
    ieee8021BridgePortMmrpFailedRegistrations  
        Counter64,  
    ieee8021BridgePortMmrpLastPduOrigin  
        MacAddress,  
    ieee8021BridgePortRestrictedGroupRegistration  
        TruthValue  
}  
  
ieee8021BridgePortMmrpEnabledStatus OBJECT-TYPE  
SYNTAX      TruthValue  
MAX-ACCESS  read-write  
STATUS      current  
DESCRIPTION  
    "The administrative state of MMRP operation on this port. The  
    value true(1) indicates that MMRP is enabled on this port  
    in all VLANs as long as ieee8021BridgeMmrpEnabledStatus is  
    also true(1). A value of false(2) indicates that MMRP is  
    disabled on this port in all VLANs: any MMRP packets received  
    will be silently discarded, and no MMRP registrations will be  
    propagated from other ports. Setting this to a value of  
    true(1) will be stored by the agent but will only take  
    effect on the MMRP protocol operation if  
    ieee8021BridgeMmrpEnabledStatus  
    also indicates the value true(1). This object affects  
    all MMRP Applicant and Registrar state machines on this  
    port. A transition from false(2) to true(1) will  
    cause a reset of all MMRP state machines on this port.  
  
    The value of this object MUST be retained across  
    reinitializations of the management system."  
DEFVAL      { true }  
::= { ieee8021BridgePortMmrpEntry 1 }  
  
ieee8021BridgePortMmrpFailedRegistrations OBJECT-TYPE  
SYNTAX      Counter64  
UNITS       "failed MMRP registrations"  
MAX-ACCESS  read-only  
STATUS      current  
DESCRIPTION  
    "The total number of failed MMRP registrations, for any  
    reason, in all VLANs, on this port."  
::= { ieee8021BridgePortMmrpEntry 2 }  
  
ieee8021BridgePortMmrpLastPduOrigin OBJECT-TYPE  
SYNTAX      MacAddress  
MAX-ACCESS  read-only  
STATUS      current  
DESCRIPTION  
    "The Source MAC Address of the last MMRP message  
    received on this port."  
::= { ieee8021BridgePortMmrpEntry 3 }  
  
ieee8021BridgePortRestrictedGroupRegistration OBJECT-TYPE  
SYNTAX      TruthValue
```

```

MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The state of Restricted Group Registration on this port.
     If the value of this control is true(1), then creation
     of a new dynamic entry is permitted only if there is a
     Static Filtering Entry for the VLAN concerned, in which
     the Registrar Administrative Control value is Normal
     Registration.

    The value of this object MUST be retained across
    reinitializations of the management system."
REFERENCE   "11.2.3.2.3, 12.11.1.3"
DEFVAL      { false }
 ::= { ieee8021BridgePortMmrpEntry 4 }

-- =====
-- I-LAN Interface configuration table
-- =====

ieee8021BridgeILanIfTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021BridgeILanIfEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "This table is a sparse augmentation of ifTable and controls
     the creation of the I-LAN Interface. An I-LAN Interface is
     used to create internal connections between bridge ports in a
     802.1 device. An I-LAN Interfaces can be directly associated
     with a set of bridge ports. An I-LAN Interfaces can also be
     used as a stacking interface to relate other interfaces before
     association to bridge ports.

    For example, an I-LAN interface can be created to link traffic
    between a PIP and a CBP. In this case a CBP is created on the
    B-Component and the CBP's related IfEntry is stacked upon the
    IfEntry of the I-LAN. The PIP is stacked upon the I-LAN using
    the IfStackTable. Finally, a VIP is created on the I-Component
    and is associated with the PIP, thus completing the path from
    the I-Component's MAC relay to the CBP on the B-Component.

    Entries in this table must be persistent over power up
    restart/reboot."
REFERENCE   "17.3.2.2"
 ::= { ieee8021BridgeInternalLan 1 }

ieee8021BridgeILanIfEntry OBJECT-TYPE
SYNTAX      Ieee8021BridgeILanIfEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "Each entry consists of a Row Status to control creation."
INDEX      { ifIndex }
 ::= { ieee8021BridgeILanIfTable 1 }

Ieee8021BridgeILanIfEntry :=
SEQUENCE {
    ieee8021BridgeILanIfRowStatus
    RowStatus
}

```

```
}

ieee8021BridgeILanIfRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object is used to create and delete entries in this
         table and the Interface table."
    ::= { ieee8021BridgeILanIfEntry 1 }

-- =====
-- 802.1D Dynamic Port Creation table
-- =====

ieee8021BridgeDot1dPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021BridgeDot1dPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table provides the capability to dynamically create and
         delete 802.1D bridge ports. Each entry in this table must
         have a corresponding entry in the ieee8021BridgeBasePortTable.

        Entries in this table must be persistent over power up
        restart/reboot."
    REFERENCE   "17.5.3"
    ::= { ieee8021BridgeDot1d 1 }

ieee8021BridgeDot1dPortEntry OBJECT-TYPE
    SYNTAX      Ieee8021BridgeDot1dPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Each entry consists of a Row Status to control creation."
    INDEX { ieee8021BridgeBasePortComponentId,
            ieee8021BridgeBasePort }
    ::= { ieee8021BridgeDot1dPortTable 1 }

Ieee8021BridgeDot1dPortEntry :=
    SEQUENCE {
        ieee8021BridgeDot1dPortRowStatus
        RowStatus
    }

ieee8021BridgeDot1dPortRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object is used to create and delete entries in this
         table and the ieee8021BridgeBasePortTable."
    ::= { ieee8021BridgeDot1dPortEntry 1 }

-- =====
-- IEEE 802.1D MIB - Conformance Information
-- =====
```

```
ieee8021BridgeCompliances
    OBJECT IDENTIFIER ::= { ieee8021BridgeConformance 1 }
ieee8021BridgeGroups
    OBJECT IDENTIFIER ::= { ieee8021BridgeConformance 2 }

-- =====
-- units of conformance
-- =====

-- =====
-- the ieee8021BridgeBase group
-- =====

ieee8021BridgeBaseBridgeGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeBaseBridgeAddress,
        ieee8021BridgeBaseNumPorts,
        ieee8021BridgeBaseComponentType
    }
    STATUS      current
    DESCRIPTION
        "Bridge level information for this device."
    ::= { ieee8021BridgeGroups 1 }

ieee8021BridgeBasePortGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeBasePortIfIndex,
        ieee8021BridgeBasePortDelayExceededDiscards,
        ieee8021BridgeBasePortMtuExceededDiscards,
        ieee8021BridgeBasePortType,
        ieee8021BridgeBasePortExternal,
        ieee8021BridgeBasePortAdminPointToPoint,
        ieee8021BridgeBasePortOperPointToPoint,
        ieee8021BridgeBasePortName
    }
    STATUS      current
    DESCRIPTION
        "Information for each port on this device."
    ::= { ieee8021BridgeGroups 2 }

ieee8021BridgeCapGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeBaseDeviceCapabilities,
        ieee8021BridgeBasePortCapabilities,
        ieee8021BridgeBasePortTypeCapabilities
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects indicating the optional
         capabilities of the device."
    ::= { ieee8021BridgeGroups 3 }

ieee8021BridgeDeviceMmrpGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeBaseMmrpEnabledStatus
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing device-level control
```

```
        for the Multicast Filtering extended bridge services."
        ::= { ieee8021BridgeGroups 4 }

-- =====
-- the ieee8021BridgeTp group
-- =====

ieee8021BridgeTpPortGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeTpPortMaxInfo,
        ieee8021BridgeTpPortInFrames,
        ieee8021BridgeTpPortOutFrames,
        ieee8021BridgeTpPortInDiscards
    }
    STATUS      current
    DESCRIPTION
        "Dynamic Filtering Database information for each port of
        the Bridge."
    ::= { ieee8021BridgeGroups 6 }

-- =====
-- Bridge Priority groups
-- =====

ieee8021BridgeDevicePriorityGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeBaseTrafficClassesEnabled
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing device-level control
        for the Priority services."
    ::= { ieee8021BridgeGroups 7 }

ieee8021BridgeDefaultPriorityGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgePortDefaultUserPriority,
        ieee8021BridgePortPriorityCodePointSelection,
        ieee8021BridgePortUseDEI,
        ieee8021BridgePortRequireDropEncoding,
        ieee8021BridgePortServiceAccessPrioritySelection
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects defining the User Priority
        applicable to each port for media that do not support
        native User Priority."
    ::= { ieee8021BridgeGroups 8 }

ieee8021BridgeRegenPriorityGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgeRegenUserPriority
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects defining the User Priorities
        applicable to each port for media that support native
        User Priority."
    ::= { ieee8021BridgeGroups 9 }
```

```
ieee8021BridgePriorityGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgePortNumTrafficClasses,
        ieee8021BridgeTrafficClass
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects defining the traffic classes
         within a bridge for each evaluated User Priority."
    ::= { ieee8021BridgeGroups 10 }

ieee8021BridgeAccessPriorityGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgePortOutboundAccessPriority
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects defining the media-dependent
         outbound access level for each priority."
    ::= { ieee8021BridgeGroups 11 }

ieee8021BridgePortMrpGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgePortMrpJoinTime,
        ieee8021BridgePortMrpLeaveTime,
        ieee8021BridgePortMrpLeaveAllTime
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing port level control
         and status information for MRP operation."
    ::= { ieee8021BridgeGroups 12 }

ieee8021BridgePortMmrpGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgePortMmrpEnabledStatus,
        ieee8021BridgePortMmrpFailedRegistrations,
        ieee8021BridgePortMmrpLastPduOrigin,
        ieee8021BridgePortRestrictedGroupRegistration
    }
    STATUS      deprecated
    DESCRIPTION
        "A collection of objects providing port level control
         and status information for MMRP operation."
    ::= { ieee8021BridgeGroups 13 }

ieee8021BridgePortDecodingGroup OBJECT-GROUP
    OBJECTS {
        ieee8021BridgePortDecodingPriority,
        ieee8021BridgePortDecodingDropEligible
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing statistics counters for
         decoding priority and drop eligibility for bridge ports."
    ::= { ieee8021BridgeGroups 14 }

ieee8021BridgePortEncodingGroup OBJECT-GROUP
```

```
OBJECTS {
    ieee8021BridgePortEncodingPriority
}
STATUS      current
DESCRIPTION
    "A collection of objects providing statistics counters for
     encoding priority and drop eligibility for bridge ports."
 ::= { ieee8021BridgeGroups 15 }

ieee8021BridgeServiceAccessPriorityGroup OBJECT-GROUP
OBJECTS {
    ieee8021BridgeServiceAccessPriorityValue
}
STATUS      current
DESCRIPTION
    "A collection of objects providing statistics
     counters for service access priority."
 ::= { ieee8021BridgeGroups 16 }

-- =====
-- Internal LAN group
-- =====

ieee8021BridgeInternalLANGroup OBJECT-GROUP
OBJECTS {
    ieee8021BridgeILanIfRowStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects providing control of internal
     LAN configuration."
 ::= { ieee8021BridgeGroups 17 }

-- =====
-- Bridge Creation Group
-- =====

ieee8021BridgeCreatableBaseBridgeGroup OBJECT-GROUP
OBJECTS {
    ieee8021BridgeBaseRowStatus
}
STATUS      current
DESCRIPTION
    "Controls the management system directed creation of
     Bridge Components."
 ::= { ieee8021BridgeGroups 18 }

-- =====
-- Dot1d Dynamic Port Creation group
-- =====

ieee8021BridgeDot1dDynamicPortCreationGroup OBJECT-GROUP
OBJECTS {
    ieee8021BridgeDot1dPortRowStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects providing dynamic creation and
     deletion of 802.1D bridge ports."
```

```
 ::= { ieee8021BridgeGroups 19 }

-- =====
-- compliance statements
-- =====

ieee8021BridgeCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting bridging
         services as defined in 802.1D-2004. Such devices support
         path cost values of 32-bits, and bridge and port priority
         values are more restricted than in 802.1D-1995.

Full support for the 802.1D management objects requires
implementation of the objects listed in the systemGroup
from the SNMPv2-MIB [RFC3418], as well as the objects
listed in the ifGeneralInformationGroup from the
IF-MIB [RFC2863]."

MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
    MANDATORY-GROUPS {
        systemGroup
    }

MODULE IF-MIB -- The interfaces MIB, RFC 2863
    MANDATORY-GROUPS {
        ifGeneralInformationGroup
    }

MODULE
    MANDATORY-GROUPS {
        ieee8021BridgeBaseBridgeGroup,
        ieee8021BridgeBasePortGroup
    }

GROUP ieee8021BridgeCreatableBaseBridgeGroup
DESCRIPTION
    "Implementation of this group is mandatory for
     bridges that allow management systems to add and delete
     bridge components. Provider Backbone Edge Bridges would
     typically fall in this category."

GROUP ieee8021BridgeTpPortGroup
DESCRIPTION
    "Implementation of this group is mandatory for
     bridges that support the transparent bridging
     mode. A transparent bridge will implement
     this group."

GROUP ieee8021BridgeInternalLANGroup
DESCRIPTION
    "Implementation of this group is optional. It can be supported
     to provide control over the relationship between interfaces and
     bridge ports where such relationships are more complex than a
     simple 1-to-1 mapping."

GROUP ieee8021BridgeDot1dDynamicPortCreationGroup
DESCRIPTION
```

"Implementation of this group is optional. It can be supported to provide the ability to dynamically create and deleted 802.1D bridge ports."

::= { ieee8021BridgeCompliances 1 }

ieee8021BridgePriorityAndMulticastFilteringCompliance MODULE-COMPLIANCE

STATUS deprecated

DESCRIPTION

"The compliance statement for device support of Priority and Multicast Filtering extended bridging services."

MODULE

MANDATORY-GROUPS { ieee8021BridgeCapGroup }

GROUP ieee8021BridgeDeviceMmrpGroup

DESCRIPTION

"This group is mandatory for devices supporting the MMRP application, defined by IEEE 802.1D Extended Filtering Services."

GROUP ieee8021BridgeDevicePriorityGroup

DESCRIPTION

"This group is mandatory only for devices supporting the priority forwarding operations defined by IEEE 802.1D."

GROUP ieee8021BridgeDefaultPriorityGroup

DESCRIPTION

"This group is mandatory only for devices supporting the priority forwarding operations defined by the extended bridge services with media types, such as Ethernet, that do not support native User Priority."

GROUP ieee8021BridgeRegenPriorityGroup

DESCRIPTION

"This group is mandatory only for devices supporting the priority forwarding operations defined by IEEE 802.1D and that have interface media types that support native User Priority, e.g., IEEE 802.5."

GROUP ieee8021BridgePriorityGroup

DESCRIPTION

"This group is mandatory only for devices supporting the priority forwarding operations defined by IEEE 802.1D."

GROUP ieee8021BridgeAccessPriorityGroup

DESCRIPTION

"This group is optional and is relevant only for devices supporting the priority forwarding operations defined by IEEE 802.1D and that have interface media types that support native Access Priority, e.g., IEEE 802.5."

GROUP ieee8021BridgePortMrpGroup

DESCRIPTION

"This group is mandatory for devices supporting any of the MRP applications: e.g., MMRP, defined by the extended filtering services of 802.1D; or MVRP, defined by 802.1Q (refer to the Q-BRIDGE-MIB for

conformance statements for MVRP)."

GROUP ieee8021BridgePortMmrpGroup

DESCRIPTION

"This group is mandatory for devices supporting the MMRP application, as defined by IEEE 802.1D Extended Filtering Services."

GROUP ieee8021BridgePortDecodingGroup

DESCRIPTION

"This group is optional and supports Priority Code Point Decoding Table for a Port of a provider bridge."

GROUP ieee8021BridgePortEncodingGroup

DESCRIPTION

"This group is optional and supports Priority Code Point Encoding Table for a Port of a provider bridge."

GROUP ieee8021BridgeServiceAccessPriorityGroup

DESCRIPTION

"This group is optional and supports Priority Code Point Encoding Table for a Port of a provider bridge."

OBJECT ieee8021BridgePortNumTrafficClasses

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required."

OBJECT ieee8021BridgeTrafficClass

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required."

OBJECT ieee8021BridgeRegenUserPriority

MIN-ACCESS read-only

DESCRIPTION

"Write access is not required."

::= { ieee8021BridgeCompliances 2 }

END

17.7.3 Definitions for the IEEE8021-SPANNING-TREE MIB module

```
IEEE8021-SPANNING-TREE-MIB DEFINITIONS ::= BEGIN

-- =====
-- MIB for IEEE 802.1D spanning tree devices
-- =====

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
    Counter64, Integer32, TimeTicks
        FROM SNMPv2-SMI
    MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
        FROM SNMPv2-CONF
    TruthValue
        FROM SNMPv2-TC
    ieee802dot1mibs, IEEE8021PbbComponentIdentifier,
    IEEE8021BridgePortNumber
        FROM IEEE8021-TC-MIB
    BridgeId, Timeout
        FROM BRIDGE-MIB
;

ieee8021SpanningTreeMib MODULE-IDENTITY
LAST-UPDATED "201103240000Z" -- March 24, 2011
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 Levi
      Postal: C/O IEEE 802.1 Working Group
              IEEE Standards Association
              445 Hoes Lane
              P.O. Box 1331
              Piscataway
              NJ 08855-1331
              USA
      E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
"The Spanning-Tree MIB module for managing devices that
support IEEE 802.1D. This MIB module is derived from the
IETF BRIDGE-MIB, RFC 4188.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."

REVISION      "201103240000Z" -- March 24, 2011
DESCRIPTION
    "Minor edits to contact information and addition of
     fragile bridge as part of 2011 revision of
     IEEE Std 802.1Q."
REVISION      "200810150000Z" -- October 15, 2008
```

```
DESCRIPTION
    "Initial revision, derived from RFC 4188."
 ::= { ieee802dot1mibs 3 }

-- =====
-- subtrees in the Spanning-Tree MIB
-- =====

ieee8021SpanningTreeNotifications
OBJECT IDENTIFIER ::= { ieee8021SpanningTreeMib 0 }

ieee8021SpanningTreeObjects
OBJECT IDENTIFIER ::= { ieee8021SpanningTreeMib 1 }

ieee8021SpanningTreeConformance
OBJECT IDENTIFIER ::= { ieee8021SpanningTreeMib 2 }

-- =====
-- the ieee8021SpanningTreeTable
-- =====

ieee8021SpanningTreeTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021SpanningTreeEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table that contains information related to STP about
     every bridge."
REFERENCE   "12.8.1"
 ::= { ieee8021SpanningTreeObjects 1 }

ieee8021SpanningTreeEntry OBJECT-TYPE
SYNTAX      Ieee8021SpanningTreeEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A list of objects containing information for each bridge
     about the Spanning Tree Protocol for that bridge."
INDEX      { ieee8021SpanningTreeComponentId }
 ::= { ieee8021SpanningTreeTable 1 }

Ieee8021SpanningTreeEntry :=
SEQUENCE {
    ieee8021SpanningTreeComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021SpanningTreeProtocolSpecification
        INTEGER,
    ieee8021SpanningTreePriority
        Integer32,
    ieee8021SpanningTreeTimeSinceTopologyChange
        TimeTicks,
    ieee8021SpanningTreeTopChanges
        Counter64,
    ieee8021SpanningTreeDesignatedRoot
        BridgeId,
    ieee8021SpanningTreeRootCost
        Integer32,
    ieee8021SpanningTreeRootPort
        IEEE8021BridgePortNumber,
```

```
ieee8021SpanningTreeMaxAge
    Timeout,
ieee8021SpanningTreeHelloTime
    Timeout,
ieee8021SpanningTreeHoldTime
    Integer32,
ieee8021SpanningTreeForwardDelay
    Timeout,
ieee8021SpanningTreeBridgeMaxAge
    Timeout,
ieee8021SpanningTreeBridgeHelloTime
    Timeout,
ieee8021SpanningTreeBridgeForwardDelay
    Timeout,
ieee8021SpanningTreeVersion
    INTEGER,
ieee8021SpanningTreeRstpTxHoldCount
    Integer32
}

ieee8021SpanningTreeComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
        multiple virtual bridge instances within a PBB. In simple
        situations where there is only a single component the default
        value is 1."
::= { ieee8021SpanningTreeEntry 1 }

ieee8021SpanningTreeProtocolSpecification OBJECT-TYPE
    SYNTAX      INTEGER {
        unknown(1),
        decLb100(2),
        ieee8021d(3),
        ieee8021q(4)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "An indication of what version of the Spanning Tree Protocol is
        being run. The value 'decLb100(2)' indicates the DEC LANbridge
        100 Spanning Tree protocol. IEEE 802.1D implementations will
        return 'ieee8021d(3)'. New enumerated values may be added in
        the future to the definition of this object to reflect future
        versions of the IEEE Spanning Tree protocol."
::= { ieee8021SpanningTreeEntry 2 }

ieee8021SpanningTreePriority OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of the write-able portion of the Bridge ID
        (i.e., the first two octets of the (8 octet long) Bridge
        ID). The other (last) 6 octets of the Bridge ID are
        given by the value of ieee8021BridgeBaseBridgeAddress.
        On bridges supporting IEEE 802.1t or IEEE 802.1w,
```

permissible values are 0-61440, in steps of 4096.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.1.1.3 a)"
 ::= { ieee8021SpanningTreeEntry 3 }

ieee8021SpanningTreeTimeSinceTopologyChange OBJECT-TYPE
 SYNTAX TimeTicks
 UNITS "centi-seconds"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The time (in hundredths of a second) since the last time a topology change was detected by the bridge entity.
 For RSTP, this reports the time since the tcWhile timer for any port on this Bridge was nonzero."
 REFERENCE "12.8.1.1.3 b)"
 ::= { ieee8021SpanningTreeEntry 4 }

ieee8021SpanningTreeTopChanges OBJECT-TYPE
 SYNTAX Counter64
 UNITS "topology changes"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The total number of topology changes detected by this bridge since the management entity was last reset or initialized.

 Discontinuities in the value of the counter can occur at re-initialization of the management system."
 REFERENCE "12.8.1.1.3 c)"
 ::= { ieee8021SpanningTreeEntry 5 }

ieee8021SpanningTreeDesignatedRoot OBJECT-TYPE
 SYNTAX BridgeId
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The bridge identifier of the root of the spanning tree, as determined by the Spanning Tree Protocol, as executed by this node. This value is used as the Root Identifier parameter in all Configuration Bridge PDUs originated by this node."
 REFERENCE "12.8.1.1.3 e)"
 ::= { ieee8021SpanningTreeEntry 6 }

ieee8021SpanningTreeRootCost OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The cost of the path to the root as seen from this bridge."
 REFERENCE "12.8.1.1.3 f)"
 ::= { ieee8021SpanningTreeEntry 7 }

```
ieee8021SpanningTreeRootPort OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The port number of the port that offers the lowest
         cost path from this bridge to the root bridge."
    REFERENCE   "12.8.1.1.3 g)"
    ::= { ieee8021SpanningTreeEntry 8 }

ieee8021SpanningTreeMaxAge OBJECT-TYPE
    SYNTAX      Timeout
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum age of Spanning Tree Protocol information
         learned from the network on any port before it is
         discarded, in units of hundredths of a second. This is
         the actual value that this bridge is currently using."
    REFERENCE   "12.8.1.1.3 h)"
    ::= { ieee8021SpanningTreeEntry 9 }

ieee8021SpanningTreeHelloTime OBJECT-TYPE
    SYNTAX      Timeout
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The amount of time between the transmission of
         Configuration bridge PDUs by this node on any port when
         it is the root of the spanning tree, or trying to become
         so, in units of hundredths of a second. This is the
         actual value that this bridge is currently using."
    REFERENCE   "12.8.1.1.3 k)"
    ::= { ieee8021SpanningTreeEntry 10 }

ieee8021SpanningTreeHoldTime OBJECT-TYPE
    SYNTAX      Integer32
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This time value determines the interval length
         during which no more than two Configuration bridge
         PDUs shall be transmitted by this node, in units
         of hundredths of a second."
    REFERENCE   "12.8.1.1.3 m)"
    ::= { ieee8021SpanningTreeEntry 11 }

ieee8021SpanningTreeForwardDelay OBJECT-TYPE
    SYNTAX      Timeout
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This time value, measured in units of hundredths of a
         second, controls how fast a port changes its spanning
         state when moving towards the Forwarding state. The
```

value determines how long the port stays in each of the Listening and Learning states, which precede the Forwarding state. This value is also used when a topology change has been detected and is underway, to age all dynamic entries in the Filtering Database.
 [Note that this value is the one that this bridge is currently using, in contrast to ieee8021SpanningTreeBridgeForwardDelay, which is the value that this bridge and all others would start using if/when this bridge were to become the root.]"

REFERENCE "12.8.1.1.3 i)"
 ::= { ieee8021SpanningTreeEntry 12 }

ieee8021SpanningTreeBridgeMaxAge OBJECT-TYPE
 SYNTAX Timeout (600..4000)
 UNITS "centi-seconds"
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The value that all bridges use for MaxAge when this bridge is acting as the root. Note that 802.1D-1998 specifies that the range for this parameter is related to the value of ieee8021SpanningTreeBridgeHelloTime. The granularity of this timer is specified by 802.1D-1998 to be 1 second. An agent may return an SNMP badValue error (or its equivalent if another protocol is used) if a set is attempted to a value that is not a whole number of seconds."

The value of this object MUST be retained across reinitializations of the management system."
 REFERENCE "12.8.1.1.3 j)"
 ::= { ieee8021SpanningTreeEntry 13 }

ieee8021SpanningTreeBridgeHelloTime OBJECT-TYPE
 SYNTAX Timeout (100..1000)
 UNITS "centi-seconds"
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The value that all bridges use for HelloTime when this bridge is acting as the root. The granularity of this timer is specified by 802.1D-1998 to be 1 second. An agent may return an SNMP badValue error (or its equivalent if another protocol is used) if a set is attempted to a value that is not a whole number of seconds."

The value of this object MUST be retained across reinitializations of the management system."
 REFERENCE "12.8.1.1.3 k)"
 ::= { ieee8021SpanningTreeEntry 14 }

ieee8021SpanningTreeBridgeForwardDelay OBJECT-TYPE
 SYNTAX Timeout (400..3000)
 UNITS "centi-seconds"
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The value that all bridges use for ForwardDelay when

this bridge is acting as the root. Note that 802.1D-1998 specifies that the range for this parameter is related to the value of ieee8021SpanningTreeBridgeMaxAge. The granularity of this timer is specified by 802.1D-1998 to be 1 second. An agent may return an SNMP badValue error (or its equivalent if another protocol is used) if a set is attempted to a value that is not a whole number of seconds.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.1.1.3 l)"
 ::= { ieee8021SpanningTreeEntry 15 }

ieee8021SpanningTreeVersion OBJECT-TYPE
SYNTAX INTEGER {
 stp(0),
 rstp(2),
 mstp(3)
}
MAX-ACCESS read-write
STATUS current
DESCRIPTION
 "The version of Spanning Tree Protocol the bridge is currently running. The values are directly from the IEEE standard. New values may be defined as future versions of the protocol become available."

The value 'stp(0)' indicates the bridge is running the Spanning Tree Protocol specified in IEEE 802.1D-1998.

The value 'rstp(2)' indicates the bridge is running the Rapid Spanning Tree Protocol specified in IEEE 802.1w and clause 17 of 802.1D-2004.

The value 'mstp(3)' indicates the bridge is running the Multiple Spanning Tree Protocol specified in IEEE802.1s and clause 13 of 802.1Q-2005.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.1.1.3 n)"
 ::= { ieee8021SpanningTreeEntry 16 }

ieee8021SpanningTreeRstpTxHoldCount OBJECT-TYPE
SYNTAX Integer32 (1..10)
MAX-ACCESS read-write
STATUS current
DESCRIPTION
 "The value used by the Port Transmit state machine to limit the maximum transmission rate. This is used by bridges that are running the Rapid Spanning Tree Protocol."

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.1.1.3 m)"
DEFVAL { 3 }
 ::= { ieee8021SpanningTreeEntry 17 }

-- =====

```
-- The Spanning Tree Port Table
-- =====

ieee8021SpanningTreePortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SpanningTreePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains port-specific information
         for the Spanning Tree Protocol."
    REFERENCE   "12.8.2"
    ::= { ieee8021SpanningTreeObjects 2 }

ieee8021SpanningTreePortEntry OBJECT-TYPE
    SYNTAX      Ieee8021SpanningTreePortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing information maintained by
         every port about the Spanning Tree Protocol state for
         that port."
    INDEX      { ieee8021SpanningTreePortComponentId,
                 ieee8021SpanningTreePort }
    ::= { ieee8021SpanningTreePortTable 1 }

Ieee8021SpanningTreePortEntry ::=
SEQUENCE {
    ieee8021SpanningTreePortComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021SpanningTreePort
        IEEE8021BridgePortNumber,
    ieee8021SpanningTreePortPriority
        Integer32,
    ieee8021SpanningTreePortState
        INTEGER,
    ieee8021SpanningTreePortEnabled
        TruthValue,
    ieee8021SpanningTreePortPathCost
        Integer32,
    ieee8021SpanningTreePortDesignatedRoot
        BridgeId,
    ieee8021SpanningTreePortDesignatedCost
        Integer32,
    ieee8021SpanningTreePortDesignatedBridge
        BridgeId,
    ieee8021SpanningTreePortDesignatedPort
        OCTET STRING,
    ieee8021SpanningTreePortForwardTransitions
        Counter64,
    ieee8021SpanningTreeRstpPortProtocolMigration
        TruthValue,
    ieee8021SpanningTreeRstpPortAdminEdgePort
        TruthValue,
    ieee8021SpanningTreeRstpPortOperEdgePort
        TruthValue,
    ieee8021SpanningTreeRstpPortAdminPathCost
        Integer32
}
```

```
ieee8021SpanningTreePortComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021SpanningTreePortEntry 1 }

ieee8021SpanningTreePort OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The port number of the port for which this entry
         contains Spanning Tree Protocol management information."
    REFERENCE   "12.8.2.1.2 a)"
    ::= { ieee8021SpanningTreePortEntry 2 }

ieee8021SpanningTreePortPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..255)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of the priority field that is contained in
         the first (in network byte order) octet of the (2 octet
         long) Port ID. The other octet of the Port ID is given
         by the value of ieee8021SpanningTreePort.
         On bridges supporting IEEE 802.1t or IEEE 802.1w,
         permissible values are 0-240, in steps of 16.

        The value of this object MUST be retained across
         reinitializations of the management system."
    REFERENCE   "12.8.2.1.3 c)"
    ::= { ieee8021SpanningTreePortEntry 3 }

ieee8021SpanningTreePortState OBJECT-TYPE
    SYNTAX      INTEGER {
                    disabled(1),
                    blocking(2),
                    listening(3),
                    learning(4),
                    forwarding(5),
                    broken(6)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The port's current state, as defined by application of
         the Spanning Tree Protocol. This state controls what
         action a port takes on reception of a frame. If the
         bridge has detected a port that is malfunctioning, it
         will place that port into the broken(6) state. For
         ports that are disabled (see
         ieee8021SpanningTreePortEnabled), this object will have a
         value of disabled(1). The values disabled, blocking,
         listening, and broken correspond to the Clause 12 port
```

state of 'Discarding'. The value learning corresponds to the Clause 12 port state of 'Learning'. The value forwarding corresponds to the Clause 12 port state of 'Forwarding'."

REFERENCE "12.8.2.1.3 b)"
 ::= { ieee8021SpanningTreePortEntry 4 }

ieee8021SpanningTreePortEnabled OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The enabled/disabled status of the port. A value of true(1) means the spanning-tree protocol is enabled for this port.

The value of this object MUST be retained across reinitializations of the management system."
 REFERENCE "12.8.2.1.3 m)"
 ::= { ieee8021SpanningTreePortEntry 5 }

ieee8021SpanningTreePortPathCost OBJECT-TYPE
 SYNTAX Integer32 (1..200000000)
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The contribution of this port to the path cost of paths towards the spanning tree root which include this port. 802.1D-1998 recommends that the default value of this parameter be in inverse proportion to the speed of the attached LAN.

The value of this object MUST be retained across reinitializations of the management system."
 REFERENCE "12.8.2.1.3 d)"
 ::= { ieee8021SpanningTreePortEntry 6 }

ieee8021SpanningTreePortDesignatedRoot OBJECT-TYPE
 SYNTAX BridgeId
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The unique Bridge Identifier of the Bridge recorded as the Root in the Configuration BPDU transmitted by the Designated Bridge for the segment to which the port is attached."
 REFERENCE "12.8.2.1.3 e)"
 ::= { ieee8021SpanningTreePortEntry 7 }

ieee8021SpanningTreePortDesignatedCost OBJECT-TYPE
 SYNTAX Integer32
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The path cost of the Designated Port of the segment connected to this port. This value is compared to the Root Path Cost field in received bridge PDUs."
 REFERENCE "12.8.2.1.3 f)"
 ::= { ieee8021SpanningTreePortEntry 8 }

```
ieee8021SpanningTreePortDesignatedBridge OBJECT-TYPE
    SYNTAX      BridgeId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Bridge Identifier of the bridge that this
         port considers to be the Designated Bridge for
         this port's segment."
    REFERENCE   "12.8.2.1.3 g)"
    ::= { ieee8021SpanningTreePortEntry 9 }

ieee8021SpanningTreePortDesignatedPort OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (2))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Port Identifier of the port on the Designated
         Bridge for this port's segment."
    REFERENCE   "12.8.2.1.3 h)"
    ::= { ieee8021SpanningTreePortEntry 10 }

ieee8021SpanningTreePortForwardTransitions OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "forwarding transitions"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of times this port has transitioned
         from the Learning state to the Forwarding state.

         Discontinuities in the value of the counter can occur
         at re-initialization of the management system, and at
         other times as indicated by the value of
         ifCounterDiscontinuityTime object of the associated
         interface (if any)."
    ::= { ieee8021SpanningTreePortEntry 11 }

ieee8021SpanningTreeRstpPortProtocolMigration OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "When operating in RSTP (version 2) mode, writing true(1)
         to this object forces this port to transmit RSTP BPDUs.
         Any other operation on this object has no effect and
         it always returns false(2) when read."
    REFERENCE   "12.8.2.5"
    ::= { ieee8021SpanningTreePortEntry 12 }

ieee8021SpanningTreeRstpPortAdminEdgePort OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the Edge Port parameter.
         A value of true(1) indicates that this port should be
         assumed as an edge-port, and a value of false(2) indicates
         that this port should be assumed as a non-edge-port.
```

Setting this object will also cause the corresponding instance of ieee8021SpanningTreeRstpPortOperEdgePort to change to the same value. Note that even when this object's value is true(1), the value of the corresponding instance of ieee8021SpanningTreeRstpPortOperEdgePort can be false(2) if a BPDU has been received.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.2.1.3 k)"
 ::= { ieee8021SpanningTreePortEntry 13 }

ieee8021SpanningTreeRstpPortOperEdgePort OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The operational value of the Edge Port parameter. The object is initialized to the value of the corresponding instance of ieee8021SpanningTreeRstpPortAdminEdgePort. When the corresponding instance of ieee8021SpanningTreeRstpPortAdminEdgePort is set, this object will be changed as well. This object will also be changed to false(2) on reception of a BPDU."
 REFERENCE "12.8.2.1.3 l)"
 ::= { ieee8021SpanningTreePortEntry 14 }

ieee8021SpanningTreeRstpPortAdminPathCost OBJECT-TYPE
 SYNTAX Integer32 (0..200000000)
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The administratively assigned value for the contribution of this port to the path cost of paths toward the spanning tree root.

Writing a value of '0' assigns the automatically calculated default Path Cost value to the port. If the default Path Cost is being used, this object returns '0' when read.

This complements the object ieee8021SpanningTreePortPathCost, which returns the operational value of the path cost.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.2.1.3 d)"
 ::= { ieee8021SpanningTreePortEntry 15 }

-- =====
-- The Spanning Tree Port Extension Table
-- =====

ieee8021SpanningTreePortExtensionTable OBJECT-TYPE
 SYNTAX SEQUENCE OF Ieee8021SpanningTreePortExtensionEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

```
"A table that contains port-specific information
for the Spanning Tree Protocol."
REFERENCE "12.8.2"
 ::= { ieee8021SpanningTreeObjects 3 }

ieee8021SpanningTreePortExtensionEntry OBJECT-TYPE
SYNTAX     Ieee8021SpanningTreePortExtensionEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
"A list of additional objects containing information
maintained by every port about the Spanning Tree
Protocol state for that port."
AUGMENTS { ieee8021SpanningTreePortEntry }
 ::= { ieee8021SpanningTreePortExtensionTable 1 }

Ieee8021SpanningTreePortExtensionEntry :=
SEQUENCE {
    ieee8021SpanningTreeRstpPortAutoEdgePort
        TruthValue,
    ieee8021SpanningTreeRstpPortAutoIsolatePort
        TruthValue,
    ieee8021SpanningTreeRstpPortIsolatePort
        TruthValue
}

ieee8021SpanningTreeRstpPortAutoEdgePort OBJECT-TYPE
SYNTAX     TruthValue
MAX-ACCESS read-write
STATUS     current
DESCRIPTION
"The administrative value of the Auto Edge Port parameter.
A value of true(1) indicates if the bridge detection state
machine (BDM, 13.31) is to detect other bridges
attached to the LAN, and set
ieee8021SpanningTreeRstpPortOperEdgePort automatically.
The default value is true(1)

This is optional and provided only by implementations
that support the automatic identification of edge ports.

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE "12.8.2.1.3 )"
 ::= { ieee8021SpanningTreePortExtensionEntry 1 }

ieee8021SpanningTreeRstpPortAutoIsolatePort OBJECT-TYPE
SYNTAX     TruthValue
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The operational value of the Isolate Port parameter.

A value of true(1) indicates a Designated Port will
transition to discarding if both
ieee8021SpanningTreeRstpPortAdminEdgePort and
ieee8021SpanningTreeRstpPortAutoEdgePort are FALSE and
the other bridge presumed to be attached to the same
```

point-to-point LAN does not transmit periodic BPDUs.

This is optional and provided only by implementations that support the automatic identification of edge ports."
REFERENCE "12.8.2.1.3"

```
::= { ieee8021SpanningTreePortExtensionEntry 2 }
```

```
ieee8021SpanningTreeRstpPortIsolatePort OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operational value of the Isolate Port parameter.

A value of true(1), Set by the bridge detection state
machine (BDM, 13.31), indicates when the Spanning Tree
Protocol Entity of a neighboring bridge has apparently
failed ."
REFERENCE "12.8.2.1.3"
::= { ieee8021SpanningTreePortExtensionEntry 3 }
```

```
-- =====
-- Notifications for use by Bridges
-- =====
-- Notifications for the Spanning Tree Protocol
-- =====
```

```
ieee8021SpanningTreeNewRoot NOTIFICATION-TYPE
    -- OBJECTS      { }
    STATUS      current
    DESCRIPTION
        "The ieee8021SpanningTreeNewRoot notification indicates that
the sending agent has become the new root of the Spanning Tree;
the notification is sent by a bridge soon after its election
as the new root, e.g., upon expiration of the Topology Change
Timer, immediately subsequent to its election."
::= { ieee8021SpanningTreeNotifications 1 }
```

```
ieee8021SpanningTreeTopologyChange NOTIFICATION-TYPE
    -- OBJECTS      { }
    STATUS      current
    DESCRIPTION
        "A ieee8021SpanningTreeTopologyChange notification is sent
by a bridge when any of its configured ports transitions from
the Learning state to the Forwarding state, or from the
Forwarding state to the Blocking state. The notification
is not sent if a ieee8021SpanningTreeNewRoot notification
is sent for the same transition."
::= { ieee8021SpanningTreeNotifications 2 }
```

```
-- =====
-- IEEE 802.1D MIB - Conformance Information
-- =====
```

```
ieee8021SpanningTreeCompliances
    OBJECT IDENTIFIER ::= { ieee8021SpanningTreeConformance 1 }
ieee8021SpanningTreeGroups
```

```
OBJECT IDENTIFIER ::= { ieee8021SpanningTreeConformance 2 }

-- =====
-- units of conformance
-- =====

-- =====
-- the ieee8021SpanningTree group
-- =====

ieee8021SpanningTreeGroup OBJECT-GROUP
OBJECTS {
    ieee8021SpanningTreeProtocolSpecification,
    ieee8021SpanningTreePriority,
    ieee8021SpanningTreeTimeSinceTopologyChange,
    ieee8021SpanningTreeTopChanges,
    ieee8021SpanningTreeDesignatedRoot,
    ieee8021SpanningTreeRootCost,
    ieee8021SpanningTreeRootPort,
    ieee8021SpanningTreeMaxAge,
    ieee8021SpanningTreeHelloTime,
    ieee8021SpanningTreeHoldTime,
    ieee8021SpanningTreeForwardDelay,
    ieee8021SpanningTreeBridgeMaxAge,
    ieee8021SpanningTreeBridgeHelloTime,
    ieee8021SpanningTreeBridgeForwardDelay,
    ieee8021SpanningTreeVersion
}
STATUS      current
DESCRIPTION
    "Bridge level Spanning Tree data for this device."
::= { ieee8021SpanningTreeGroups 1 }

ieee8021SpanningTreeRstpGroup OBJECT-GROUP
OBJECTS {
    ieee8021SpanningTreeRstpTxHoldCount
}
STATUS      current
DESCRIPTION
    "Bridge level Rstp data for this device."
::= { ieee8021SpanningTreeGroups 2 }

ieee8021SpanningTreePortGroup OBJECT-GROUP
OBJECTS {
    ieee8021SpanningTreePortPriority,
    ieee8021SpanningTreePortState,
    ieee8021SpanningTreePortEnabled,
    ieee8021SpanningTreePortPathCost,
    ieee8021SpanningTreePortDesignatedRoot,
    ieee8021SpanningTreePortDesignatedCost,
    ieee8021SpanningTreePortDesignatedBridge,
    ieee8021SpanningTreePortDesignatedPort,
    ieee8021SpanningTreePortForwardTransitions
}
STATUS      current
DESCRIPTION
    "Spanning Tree data for each port on this device."
::= { ieee8021SpanningTreeGroups 3 }
```

```

ieee8021SpanningTreeRstpPortGroup OBJECT-GROUP
OBJECTS {
    ieee8021SpanningTreeRstpPortProtocolMigration,
    ieee8021SpanningTreeRstpPortAdminEdgePort,
    ieee8021SpanningTreeRstpPortOperEdgePort,
    ieee8021SpanningTreeRstpPortAdminPathCost
}
STATUS      current
DESCRIPTION
    "Rstp data for each port on this device."
::= { ieee8021SpanningTreeGroups 4 }

ieee8021SpanningTreeRstpFragileGroup OBJECT-GROUP
OBJECTS {
    ieee8021SpanningTreeRstpPortAutoEdgePort,
    ieee8021SpanningTreeRstpPortAutoIsolatePort,
    ieee8021SpanningTreeRstpPortIsolatePort
}
STATUS      current
DESCRIPTION
    "Rstp fragile bridge data for each port
     on this device."
::= { ieee8021SpanningTreeGroups 6 }

-- =====
-- The Notification Group
-- =====

ieee8021SpanningTreeNotificationGroup NOTIFICATION-GROUP
NOTIFICATIONS {
    ieee8021SpanningTreeNewRoot,
    ieee8021SpanningTreeTopologyChange
}
STATUS      current
DESCRIPTION
    "Group of notifications."
::= { ieee8021SpanningTreeGroups 5 }

-- =====
-- compliance statements
-- =====

ieee8021SpanningTreeCompliance MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for devices supporting the
     Spanning Tree Protocol."

MODULE
MANDATORY-GROUPS {
    ieee8021SpanningTreeGroup,
    ieee8021SpanningTreePortGroup
}

OBJECT ieee8021SpanningTreePriority
SYNTAX Integer32 (0|4096|8192|12288|16384|20480|24576
                  |28672|32768|36864|40960|45056|49152
                  |53248|57344|61440)
DESCRIPTION

```

"The possible values defined by IEEE 802.1t."

OBJECT ieee8021SpanningTreePortPriority
SYNTAX Integer32 (0|16|32|48|64|80|96|112|128
|144|160|176|192|208|224|240)
DESCRIPTION
"The possible values defined by IEEE 802.1t."

GROUP ieee8021SpanningTreeNotificationGroup
DESCRIPTION
"Implementation of this group is optional."

 ::= { ieee8021SpanningTreeCompliances 1 }

ieee8021SpanningTreeRstpCompliance MODULE-COMPLIANCE
STATUS current
DESCRIPTION
"The compliance statement for devices supporting the
Rapid Spanning Tree Protocol."

MODULE
MANDATORY-GROUPS {
 ieee8021SpanningTreeGroup,
 ieee8021SpanningTreeRstpGroup,
 ieee8021SpanningTreePortGroup,
 ieee8021SpanningTreeRstpPortGroup
}

OBJECT ieee8021SpanningTreePriority
SYNTAX Integer32 (0|4096|8192|12288|16384|20480|24576
|28672|32768|36864|40960|45056|49152
|53248|57344|61440)
DESCRIPTION
"The possible values defined by IEEE 802.1t."

OBJECT ieee8021SpanningTreePortPriority
SYNTAX Integer32 (0|16|32|48|64|80|96|112|128
|144|160|176|192|208|224|240)
DESCRIPTION
"The possible values defined by IEEE 802.1t."

GROUP ieee8021SpanningTreeNotificationGroup
DESCRIPTION
"Implementation of this group is optional."

GROUP ieee8021SpanningTreeRstpFragileGroup
DESCRIPTION
"Implementation of this group is optional."

 ::= { ieee8021SpanningTreeCompliances 2 }

END

17.7.4 Definitions for the IEEE8021-Q-BRIDGE MIB module

```
IEEE8021-Q-BRIDGE-MIB DEFINITIONS ::= BEGIN

-- =====
-- MIB for IEEE 802.1Q Devices
-- =====

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Gauge32,
    Counter64, Unsigned32, TimeTicks, Integer32
        FROM SNMPv2-SMI
    RowStatus, StorageType, TruthValue, MacAddress
        FROM SNMPv2-TC
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    ieee8021BridgeBasePortComponentId, ieee8021BridgeBasePort,
    ieee8021BridgeBasePortEntry
        FROM IEEE8021-BRIDGE-MIB
    ieee802dot1mibs, IEEE8021PbbComponentIdentifier,
    IEEE8021BridgePortNumber, IEEE8021BridgePortNumberOrZero,
    IEEE8021VlanIndex, IEEE8021VlanIndexOrWildcard,
    IEEE8021PortAcceptableFrameTypes
        FROM IEEE8021-TC-MIB
    PortList, VlanId
        FROM Q-BRIDGE-MIB
    TimeFilter
        FROM RMON2-MIB;

ieee8021QBridgeMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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 Levi
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
    "The VLAN Bridge MIB module for managing Virtual Bridged
     Local Area Networks, as defined by IEEE 802.1Q-2011.

This MIB module is derived from the IETF Q-BRIDGE-MIB
from RFC 4363.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2010.
```

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
REVISION "201102270000Z" -- February 27, 2011
DESCRIPTION
"Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."

REVISION "200810150000Z" -- October 15, 2008
DESCRIPTION
"Initial version, derived from RFC 4363."
 ::= { ieee802dot1mibs 4 }

ieee8021QBridgeMibObjects OBJECT IDENTIFIER ::= { ieee8021QBridgeMib 1 }

-- =====
-- subtrees in the Q-BRIDGE MIB
-- =====

ieee8021QBridgeBase OBJECT IDENTIFIER ::= { ieee8021QBridgeMibObjects 1 }
ieee8021QBridgeTp OBJECT IDENTIFIER ::= { ieee8021QBridgeMibObjects 2 }
ieee8021QBridgeStatic OBJECT IDENTIFIER ::= { ieee8021QBridgeMibObjects 3 }
ieee8021QBridgeVlan OBJECT IDENTIFIER ::= { ieee8021QBridgeMibObjects 4 }
ieee8021QBridgeProtocol OBJECT IDENTIFIER ::= { ieee8021QBridgeMibObjects 5 }

-- =====
-- ieee8021QBridgeBase subtree
-- =====

-- =====
-- ieee8021QBridgeTable - Table of VLAN bridges
-- =====

ieee8021QBridgeTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021QBridgeEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"A table that contains generic information about every
VLAN bridge."
REFERENCE "12.4"
 ::= { ieee8021QBridgeBase 1 }

ieee8021QBridgeEntry OBJECT-TYPE
SYNTAX Ieee8021QBridgeEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"A list of objects containing information for each VLAN bridge."
INDEX { ieee8021QBridgeComponentId }
 ::= { ieee8021QBridgeTable 1 }

Ieee8021QBridgeEntry ::=
SEQUENCE {
 ieee8021QBridgeComponentId IEEE8021PbbComponentIdentifier,
 ieee8021QBridgeVlanVersionNumber INTEGER,
 ieee8021QBridgeMaxVlanId VlanId,
 ieee8021QBridgeMaxSupportedVlans Unsigned32,
 ieee8021QBridgeNumVlans Gauge32,
}

```

        ieee8021QBridgeMvrpEnabledStatus  TruthValue
    }

ieee8021QBridgeComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021QBridgeEntry 1 }

ieee8021QBridgeVlanVersionNumber OBJECT-TYPE
    SYNTAX      INTEGER {
        version1(1),
        version2(2)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The version number of IEEE 802.1Q that this device
         supports. Reported as 1 by VLAN Bridges that support
         only SST operation, and reported as 2 by VLAN Bridges
         that support MST operation."
    REFERENCE   "12.10.1.1"
    ::= { ieee8021QBridgeEntry 2 }

ieee8021QBridgeMaxVlanId OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum IEEE 802.1Q VLAN-ID that this device
         supports."
    REFERENCE   "9.6"
    ::= { ieee8021QBridgeEntry 3 }

ieee8021QBridgeMaxSupportedVlans OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS      "vlans"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum number of IEEE 802.1Q VLANs that this
         device supports."
    REFERENCE   "12.10.1.1"
    ::= { ieee8021QBridgeEntry 4 }

ieee8021QBridgeNumVlans OBJECT-TYPE
    SYNTAX      Gauge32
    UNITS      "vlans"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current number of IEEE 802.1Q VLANs that are
         configured in this device."
    REFERENCE   "12.7.1.1"

```

```
 ::= { ieee8021QBridgeEntry 5 }

ieee8021QBridgeMvrpEnabledStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS     current
    DESCRIPTION
        "The administrative status requested by management for
         MVRP. The value true(1) indicates that MVRP should
         be enabled on this device, on all ports for which it has
         not been specifically disabled. When false(2), MVRP
         is disabled on all ports, and all MVRP packets will be
         forwarded transparently. This object affects all MVRP
         Applicant and Registrar state machines. A transition
         from false(2) to true(1) will cause a reset of all
         MVRP state machines on all ports.

        The value of this object MUST be retained across
        reinitializations of the management system."
    DEFVAL    { true }
    ::= { ieee8021QBridgeEntry 6 }

-- =====
-- ieee8021QBridgeCVlanPortTable - Table of C-VLAN ports
-- =====

ieee8021QBridgeCVlanPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeCVlanPortEntry
    MAX-ACCESS  not-accessible
    STATUS     current
    DESCRIPTION
        "This table provides the capability to create and delete
         customer VLAN ports. Entries in this table must be
         persistent over power up restart/reboot."
    REFERENCE  "12.16.1.1.3 h4), 12.16.2.1/2,
                12.13.1.1, 12.13.1.2, 12.15.2.1, 12.15.2.2"
    ::= { ieee8021QBridgeBase 2 }

Ieee8021QBridgeCVlanPortEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeCVlanPortEntry
    MAX-ACCESS  not-accessible
    STATUS     current
    DESCRIPTION
        "A list of objects containing information for each VLAN bridge."
    INDEX    { ieee8021QBridgeCVlanPortComponentId,
              ieee8021QBridgeCVlanPortNumber }
    ::= { ieee8021QBridgeCVlanPortTable 1 }

Ieee8021QBridgeCVlanPortEntry ::=
SEQUENCE {
    ieee8021QBridgeCVlanPortComponentId IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeCVlanPortNumber      IEEE8021BridgePortNumber,
    ieee8021QBridgeCVlanPortRowStatus   RowStatus
}

ieee8021QBridgeCVlanPortComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS     current
```

```

DESCRIPTION
    "The component containing the customer VLAN port represented
     by this row."
 ::= { ieee8021QBridgeCVlanPortEntry 1 }

ieee8021QBridgeCVlanPortNumber OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The customer VLAN port number represented by this row."
 ::= { ieee8021QBridgeCVlanPortEntry 2 }

ieee8021QBridgeCVlanPortRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This indicates the status of the entry, and is used to create
         and delete entries in this table. Each entry in this table that
         is valid will have a corresponding entry in the
         ieee8021BridgeBasePortTable whose value for
         ieee8021BridgeBasePortType is customerVlanPort(2). The
         corresponding value of ieee8021BridgeBasePortIfIndex must
         be set at the time the value of this object transitions
         to valid(1).

        Entries in this table must be persistent across
        reinitializations of the management system."
 ::= { ieee8021QBridgeCVlanPortEntry 3 }

-- =====
-- the ieee8021QBridgeTp subtree
-- =====

-- =====
-- the current Filtering Database Table
-- =====

ieee8021QBridgeFdbTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeFdbEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains configuration and control
         information for each Filtering Database currently
         operating on this device. Entries in this table appear
         automatically when VLANs are assigned FDB IDs in the
         ieee8021QBridgeVlanCurrentTable."
    REFERENCE   "12.7.1"
 ::= { ieee8021QBridgeTp 1 }

ieee8021QBridgeFdbEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeFdbEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a specific Filtering Database."
    INDEX      { ieee8021QBridgeFdbComponentId,

```

```
        ieee8021QBridgeFdbId }
 ::= { ieee8021QBridgeFdbTable 1 }

Ieee8021QBridgeFdbEntry ::=

SEQUENCE {
    ieee8021QBridgeFdbComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeFdbId
        Unsigned32,
    ieee8021QBridgeFdbDynamicCount
        Gauge32,
    ieee8021QBridgeFdbLearnedEntryDiscards
        Counter64,
    ieee8021QBridgeFdbAgingTime
        Integer32
}

ieee8021QBridgeFdbComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
 ::= { ieee8021QBridgeFdbEntry 1 }

ieee8021QBridgeFdbId OBJECT-TYPE
    SYNTAX      Unsigned32 (0..4294967295)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The identity of this Filtering Database."
 ::= { ieee8021QBridgeFdbEntry 2 }

ieee8021QBridgeFdbDynamicCount OBJECT-TYPE
    SYNTAX      Gauge32
    UNITS      "database entries"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current number of dynamic entries in this
         Filtering Database."
    REFERENCE   "12.7.1.1.3"
 ::= { ieee8021QBridgeFdbEntry 3 }

ieee8021QBridgeFdbLearnedEntryDiscards OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "database entries"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The total number of Filtering Database entries that
         have been or would have been learned, but have been
         discarded due to a lack of storage space in the
         Filtering Database. If this counter is increasing, it
         indicates that the Filtering Database is regularly
         becoming full (a condition that has unpleasant
```

performance effects on the subnetwork). If this counter has a significant value but is not presently increasing, it indicates that the problem has been occurring but is not persistent.

Discontinuities in the value of the counter can occur at re-initialization of the management system."

```
::= { ieee8021QBridgeFdbEntry 4 }
```

```
ieee8021QBridgeFdbAgingTime OBJECT-TYPE
    SYNTAX      Integer32 (10..1000000)
    UNITS      "seconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The timeout period in seconds for aging out
         dynamically-learned forwarding information.
         802.1D-1998 recommends a default of 300 seconds.
```

The value of this object MUST be retained across reinitializations of the management system."

```
REFERENCE    "12.7.1.2"
```

```
::= { ieee8021QBridgeFdbEntry 5 }
```

```
-- =====
-- Multiple Filtering Databases for 802.1Q Transparent Devices
-- This table is an alternative to the ieee8021BridgeTpFdbTable,
-- previously defined for 802.1D devices that only support a
-- single Filtering Database.
-- =====
```

```
ieee8021QBridgeTpFdbTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeTpFdbEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains information about unicast entries
         for which the device has forwarding and/or filtering
         information. This information is used by the
         transparent bridging function in determining how to
         propagate a received frame."
    REFERENCE    "12.7.1"
    ::= { ieee8021QBridgeTp 2 }
```

```
ieee8021QBridgeTpFdbEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeTpFdbEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about a specific unicast MAC address for
         which the device has some forwarding and/or filtering
         information."
    INDEX      { ieee8021QBridgeFdbComponentId,
                 ieee8021QBridgeFdbId,
                 ieee8021QBridgeTpFdbAddress }
    ::= { ieee8021QBridgeTpFdbTable 1 }
```

```
Ieee8021QBridgeTpFdbEntry ::=
SEQUENCE {
```

```
ieee8021QBridgeTpFdbAddress
    MacAddress,
ieee8021QBridgeTpFdbPort
    IEEE8021BridgePortNumberOrZero,
ieee8021QBridgeTpFdbStatus
    INTEGER
}

ieee8021QBridgeTpFdbAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A unicast MAC address for which the device has
    forwarding and/or filtering information."
 ::= { ieee8021QBridgeTpFdbEntry 1 }

ieee8021QBridgeTpFdbPort OBJECT-TYPE
SYNTAX      IEEE8021BridgePortNumberOrZero
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Either the value '0', or the port number of the port on
    which a frame having a source address equal to the value
    of the corresponding instance of ieee8021QBridgeTpFdbAddress has
    been seen. A value of '0' indicates that the port
    number has not been learned but that the device does
    have some forwarding/filtering information about this
    address (e.g., in the ieee8021QBridgeStaticUnicastTable).
    Implementors are encouraged to assign the port value to
    this object whenever it is learned, even for addresses
    for which the corresponding value of ieee8021QBridgeTpFdbStatus is
    not learned(3)."
 ::= { ieee8021QBridgeTpFdbEntry 2 }

ieee8021QBridgeTpFdbStatus OBJECT-TYPE
SYNTAX      INTEGER {
    other(1),
    invalid(2),
    learned(3),
    self(4),
    mgmt(5)
}
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The status of this entry. The meanings of the values
    are:
        other(1) - none of the following. This may include
            the case where some other MIB object (not the
            corresponding instance of ieee8021QBridgeTpFdbPort, nor an
            entry in the ieee8021QBridgeStaticUnicastTable) is being
            used to determine if and how frames addressed to
            the value of the corresponding instance of
            ieee8021QBridgeTpFdbAddress are being forwarded.
        invalid(2) - this entry is no longer valid (e.g., it
            was learned but has since aged out), but has not
            yet been flushed from the table.
        learned(3) - the value of the corresponding instance
```

```

        of ieee8021QBridgeTpFdbPort was learned and is being used.
        self(4) - the value of the corresponding instance of
        ieee8021QBridgeTpFdbAddress represents one of the device's
        addresses. The corresponding instance of
        ieee8021QBridgeTpFdbPort indicates which of the device's
        ports has this address.
        mgmt(5) - the value of the corresponding instance of
        ieee8021QBridgeTpFdbAddress is also the value of an
        existing instance of ieee8021QBridgeStaticUnicastAddress."
 ::= { ieee8021QBridgeTpFdbEntry 3 }

-- =====
-- Dynamic Group Registration Table
-- =====

ieee8021QBridgeTpGroupTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeTpGroupEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing filtering information for VLANs
        configured into the bridge by (local or network)
        management, or learned dynamically, specifying the set of
        ports to which frames received on a VLAN for this FDB
        and containing a specific Group destination address are
        allowed to be forwarded."
    REFERENCE   "12.7.4"
 ::= { ieee8021QBridgeTp 3 }

Ieee8021QBridgeTpGroupEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeTpGroupEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Filtering information configured into the bridge by
        management, or learned dynamically, specifying the set of
        ports to which frames received on a VLAN and containing
        a specific Group destination address are allowed to be
        forwarded. The subset of these ports learned dynamically
        is also provided."
    INDEX      { ieee8021QBridgeVlanCurrentComponentId,
                ieee8021QBridgeVlanIndex,
                ieee8021QBridgeTpGroupAddress }
 ::= { ieee8021QBridgeTpGroupTable 1 }

Ieee8021QBridgeTpGroupEntry ::=
SEQUENCE {
    ieee8021QBridgeTpGroupAddress
        MacAddress,
    ieee8021QBridgeTpGroupEgressPorts
        PortList,
    ieee8021QBridgeTpGroupLearnt
        PortList
}
}

ieee8021QBridgeTpGroupAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  not-accessible
    STATUS      current

```

DESCRIPTION
"The destination Group MAC address in a frame to which this entry's filtering information applies."
 ::= { ieee8021QBridgeTpGroupEntry 1 }

ieee8021QBridgeTpGroupEgressPorts OBJECT-TYPE
SYNTAX PortList
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The complete set of ports, in this VLAN, to which frames destined for this Group MAC address are currently being explicitly forwarded. This does not include ports for which this address is only implicitly forwarded, in the ieee8021QBridgeForwardAllPorts list."
 ::= { ieee8021QBridgeTpGroupEntry 2 }

ieee8021QBridgeTpGroupLearned OBJECT-TYPE
SYNTAX PortList
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The subset of ports in ieee8021QBridgeTpGroupEgressPorts that were learned by MMRP or some other dynamic mechanism, in this Filtering database."
 ::= { ieee8021QBridgeTpGroupEntry 3 }

-- =====
-- Service Requirements subtree
-- =====

ieee8021QBridgeForwardAllTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021QBridgeForwardAllEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"A table containing forwarding information for each VLAN, specifying the set of ports to which forwarding of all multicasts applies, configured statically by management or dynamically by MMRP. An entry appears in this table for all VLANs that are currently instantiated."
REFERENCE "12.7.2, 12.7.7"
 ::= { ieee8021QBridgeTp 4 }

ieee8021QBridgeForwardAllEntry OBJECT-TYPE
SYNTAX Ieee8021QBridgeForwardAllEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"Forwarding information for a VLAN, specifying the set of ports to which all multicasts should be forwarded, configured statically by management or dynamically by MMRP."
INDEX { ieee8021QBridgeVlanCurrentComponentId,
 ieee8021QBridgeForwardAllVlanIndex }
 ::= { ieee8021QBridgeForwardAllTable 1 }

Ieee8021QBridgeForwardAllEntry ::=

```

SEQUENCE {
    ieee8021QBridgeForwardAllVlanIndex
        IEEE8021VlanIndexOrWildcard,
    ieee8021QBridgeForwardAllPorts
        PortList,
    ieee8021QBridgeForwardAllStaticPorts
        PortList,
    ieee8021QBridgeForwardAllForbiddenPorts
        PortList
}

ieee8021QBridgeForwardAllVlanIndex OBJECT-TYPE
SYNTAX      IEEE8021VlanIndexOrWildcard
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The VLAN-ID or other identifier referring to this VLAN."
::= { ieee8021QBridgeForwardAllEntry 1 }

ieee8021QBridgeForwardAllPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The complete set of ports in this VLAN to which all
     multicast group-addressed frames are to be forwarded.
     This includes ports for which this need has been
     determined dynamically by MMRP, or configured statically
     by management."
::= { ieee8021QBridgeForwardAllEntry 2 }

ieee8021QBridgeForwardAllStaticPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The set of ports configured by management in this VLAN
     to which all multicast group-addressed frames are to be
     forwarded. Ports entered in this list will also appear
     in the complete set shown by ieee8021QBridgeForwardAllPorts. This
     value will be restored after the device is reset. This
     only applies to ports that are members of the VLAN,
     defined by ieee8021QBridgeVlanCurrentEgressPorts. A port may not
     be added in this set if it is already a member of the
     set of ports in ieee8021QBridgeForwardAllForbiddenPorts. The
     default value is a string of ones of appropriate length,
     to indicate the standard behaviour of using basic
     filtering services, i.e., forward all multicasts to all
     ports.

    The value of this object MUST be retained across
     reinitializations of the management system."
::= { ieee8021QBridgeForwardAllEntry 3 }

ieee8021QBridgeForwardAllForbiddenPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION

```

"The set of ports configured by management in this VLAN for which the Service Requirement attribute Forward All Multicast Groups may not be dynamically registered by MMRP. This value will be restored after the device is reset. A port may not be added in this set if it is already a member of the set of ports in ieee8021QBridgeForwardAllStaticPorts. The default value is a string of zeros of appropriate length.

The value of this object MUST be retained across reinitializations of the management system."

::= { ieee8021QBridgeForwardAllEntry 4 }

ieee8021QBridgeForwardUnregisteredTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021QBridgeForwardUnregisteredEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"A table containing forwarding information for each VLAN, specifying the set of ports to which forwarding of multicast group-addressed frames for which no more specific forwarding information applies. This is configured statically by management and determined dynamically by MMRP. An entry appears in this table for all VLANs that are currently instantiated."
REFERENCE "12.7.2, 12.7.7"
 ::= { ieee8021QBridgeTp 5 }

ieee8021QBridgeForwardUnregisteredEntry OBJECT-TYPE
SYNTAX Ieee8021QBridgeForwardUnregisteredEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"Forwarding information for a VLAN, specifying the set of ports to which all multicasts for which there is no more specific forwarding information shall be forwarded. This is configured statically by management or dynamically by MMRP."
INDEX { ieee8021QBridgeVlanCurrentComponentId,
 ieee8021QBridgeForwardUnregisteredVlanIndex }
 ::= { ieee8021QBridgeForwardUnregisteredTable 1 }

Ieee8021QBridgeForwardUnregisteredEntry ::=
SEQUENCE {
 ieee8021QBridgeForwardUnregisteredVlanIndex
 IEEE8021VlanIndexOrWildcard,
 ieee8021QBridgeForwardUnregisteredPorts
 PortList,
 ieee8021QBridgeForwardUnregisteredStaticPorts
 PortList,
 ieee8021QBridgeForwardUnregisteredForbiddenPorts
 PortList
}

ieee8021QBridgeForwardUnregisteredVlanIndex OBJECT-TYPE
SYNTAX IEEE8021VlanIndexOrWildcard
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION

```
"The VLAN-ID or other identifier referring to this VLAN."  
 ::= { ieee8021QBridgeForwardUnregisteredEntry 1 }

ieee8021QBridgeForwardUnregisteredPorts OBJECT-TYPE
    SYNTAX      PortList
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The complete set of ports in this VLAN to which
         multicast group-addressed frames for which there is no
         more specific forwarding information will be forwarded.
         This includes ports for which this need has been
         determined dynamically by MMRP, or configured statically
         by management."
 ::= { ieee8021QBridgeForwardUnregisteredEntry 2 }

ieee8021QBridgeForwardUnregisteredStaticPorts OBJECT-TYPE
    SYNTAX      PortList
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The set of ports configured by management, in this
         VLAN, to which multicast group-addressed frames for
         which there is no more specific forwarding information
         are to be forwarded. Ports entered in this list will
         also appear in the complete set shown by
         ieee8021QBridgeForwardUnregisteredPorts. This value will be
         restored after the device is reset. A port may not be
         added in this set if it is already a member of the set
         of ports in ieee8021QBridgeForwardUnregisteredForbiddenPorts. The
         default value is a string of zeros of appropriate
         length, although this has no effect with the default
         value of ieee8021QBridgeForwardAllStaticPorts.

The value of this object MUST be retained across
reinitializations of the management system."
 ::= { ieee8021QBridgeForwardUnregisteredEntry 3 }

ieee8021QBridgeForwardUnregisteredForbiddenPorts OBJECT-TYPE
    SYNTAX      PortList
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The set of ports configured by management in this VLAN
         for which the Service Requirement attribute Forward
         Unregistered Multicast Groups may not be dynamically
         registered by MMRP. This value will be restored after
         the device is reset. A port may not be added in this
         set if it is already a member of the set of ports in
         ieee8021QBridgeForwardUnregisteredStaticPorts. The default value
         is a string of zeros of appropriate length.

The value of this object MUST be retained across
reinitializations of the management system."
 ::= { ieee8021QBridgeForwardUnregisteredEntry 4 }

-- =====
-- The Static (Destination-Address Filtering) Database
-- =====
```

```
ieee8021QBridgeStaticUnicastTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeStaticUnicastEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing filtering information for Unicast
         MAC addresses for each Filtering Database, configured
         into the device by (local or network) management
         specifying the set of ports to which frames received
         from specific ports and containing specific unicast
         destination addresses are allowed to be forwarded.
         Entries are valid for unicast addresses only."
```

Two modes of operation are supported by this table. When the receive port index is non-zero, this table is supporting an 802.1D filtering database as specified in 14.7.6.1. If the receive port is zero, the table is operating as specified in 802.1Q 8.8.1 and 12.7.7. An agent must at least support the 802.1Q mode of operation."

REFERENCE "802.1D 7.9.1, 14.7.6.1;
 802.1Q 12.7.7, 8.8.1"

::= { ieee8021QBridgeStatic 1 }

```
ieee8021QBridgeStaticUnicastEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeStaticUnicastEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Filtering information configured into the device by
         (local or network) management specifying the set of
         ports to which frames received from a specific port and
         containing a specific unicast destination address are
         allowed to be forwarded."
    INDEX      {
        ieee8021QBridgeStaticUnicastComponentId,
        ieee8021QBridgeStaticUnicastVlanIndex,
        ieee8021QBridgeStaticUnicastAddress,
        ieee8021QBridgeStaticUnicastReceivePort
    }
    ::= { ieee8021QBridgeStaticUnicastTable 1 }
```

```
Ieee8021QBridgeStaticUnicastEntry ::=
SEQUENCE {
    ieee8021QBridgeStaticUnicastComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeStaticUnicastVlanIndex
        IEEE8021VlanIndexOrWildcard,
    ieee8021QBridgeStaticUnicastAddress
        MacAddress,
    ieee8021QBridgeStaticUnicastReceivePort
        IEEE8021BridgePortNumberOrZero,
    ieee8021QBridgeStaticUnicastStaticEgressPorts
        PortList,
    ieee8021QBridgeStaticUnicastForbiddenEgressPorts
        PortList,
    ieee8021QBridgeStaticUnicastStorageType
        StorageType,
```

```
ieee8021QBridgeStaticUnicastRowStatus
    RowStatus
}

ieee8021QBridgeStaticUnicastComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021QBridgeStaticUnicastEntry 1 }

ieee8021QBridgeStaticUnicastVlanIndex OBJECT-TYPE
    SYNTAX      IEEE8021VlanIndexOrWildcard
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Vlan to which this entry applies."
    ::= { ieee8021QBridgeStaticUnicastEntry 2 }

ieee8021QBridgeStaticUnicastAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The destination MAC address in a frame to which this
         entry's filtering information applies. This object must
         take the value of a unicast address."
    ::= { ieee8021QBridgeStaticUnicastEntry 3 }

ieee8021QBridgeStaticUnicastReceivePort OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumberOrZero
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Either the value '0' or the port number of the port
         from which a frame must be received in order for this
         entry's filtering information to apply. A value of zero
         indicates that this entry applies on all ports of the
         device for which there is no other applicable entry. An
         implementation is required to support the '0' value and
         may optionally support non-zero values for this column."
    ::= { ieee8021QBridgeStaticUnicastEntry 4 }

ieee8021QBridgeStaticUnicastStaticEgressPorts OBJECT-TYPE
    SYNTAX      PortList
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The set of ports to which frames received from a
         specific port and destined for a specific unicast address
         must be forwarded, regardless of
         any dynamic information, e.g., from MMRP. A port may not
         be added in this set if it is already a member of the
         set of ports in ieee8021QBridgeStaticUnicastForbiddenEgressPorts.
         The default value of this object is a string of ones of
```

```
    appropriate length."
DEFVAL      { ''H }
 ::= { ieee8021QBridgeStaticUnicastEntry 5 }

ieee8021QBridgeStaticUnicastForbiddenEgressPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The set of ports to which frames received from a
specific port and destined for a specific unicast
MAC address must not be forwarded, regardless
of any dynamic information, e.g., from MMRP. A port may
not be added in this set if it is already a member of the
set of ports in ieee8021QBridgeStaticUnicastStaticEgressPorts.
The default value of this object is a string of zeros of
appropriate length."
DEFVAL      { ''H }
 ::= { ieee8021QBridgeStaticUnicastEntry 6 }

ieee8021QBridgeStaticUnicastStorageType OBJECT-TYPE
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The storage type for this conceptual row. If this object
has a value of permanent(4), then no other objects are
required to be able to be modified."
DEFVAL { nonVolatile }
 ::= { ieee8021QBridgeStaticUnicastEntry 7 }

ieee8021QBridgeStaticUnicastRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"This object indicates the status of this entry, and is used
to create/delete entries in the table.

An entry of this table may be set to active without setting
any other columns of the table. Also, other columns of this
table may be set while the value of this object is active(1)."
 ::= { ieee8021QBridgeStaticUnicastEntry 8 }

ieee8021QBridgeStaticMulticastTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021QBridgeStaticMulticastEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table containing filtering information for Multicast
and Broadcast MAC addresses for each VLAN, configured
into the device by (local or network) management
specifying the set of ports to which frames received
from specific ports and containing specific Multicast
and Broadcast destination addresses are allowed to be
forwarded. A value of zero in this table (as the port
number from which frames with a specific destination
address are received) is used to specify all ports for
```

which there is no specific entry in this table for that particular destination address. Entries are valid for Multicast and Broadcast addresses only."

REFERENCE "12.7.7, 8.8.1"

::= { ieee8021QBridgeStatic 2 }

```

ieee8021QBridgeStaticMulticastEntry OBJECT-TYPE
SYNTAX      Ieee8021QBridgeStaticMulticastEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
  "Filtering information configured into the device by
  (local or network) management specifying the set of
  ports to which frames received from this specific port
  for this VLAN and containing this Multicast or Broadcast
  destination address are allowed to be forwarded."
INDEX   {
  ieee8021QBridgeVlanCurrentComponentId,
  ieee8021QBridgeVlanIndex,
  ieee8021QBridgeStaticMulticastAddress,
  ieee8021QBridgeStaticMulticastReceivePort
}
 ::= { ieee8021QBridgeStaticMulticastTable 1 }

Ieee8021QBridgeStaticMulticastEntry ::=
SEQUENCE {
  ieee8021QBridgeStaticMulticastAddress
    MacAddress,
  ieee8021QBridgeStaticMulticastReceivePort
    IEEE8021BridgePortNumberOrZero,
  ieee8021QBridgeStaticMulticastStaticEgressPorts
    PortList,
  ieee8021QBridgeStaticMulticastForbiddenEgressPorts
    PortList,
  ieee8021QBridgeStaticMulticastStorageType
    StorageType,
  ieee8021QBridgeStaticMulticastRowStatus
    RowStatus
}

```

ieee8021QBridgeStaticMulticastAddress OBJECT-TYPE

SYNTAX MacAddress

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The destination MAC address in a frame to which this entry's filtering information applies. This object must take the value of a Multicast or Broadcast address."

::= { ieee8021QBridgeStaticMulticastEntry 1 }

```

ieee8021QBridgeStaticMulticastReceivePort OBJECT-TYPE
SYNTAX      IEEE8021BridgePortNumberOrZero
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
  "Either the value '0' or the port number of the port
  from which a frame must be received in order for this
  entry's filtering information to apply. A value of zero
  indicates that this entry applies on all ports of the

```

```
device for which there is no other applicable entry. An
implementation is required to support the '0' value and
may optionally support non-zero values for this column."
 ::= { ieee8021QBridgeStaticMulticastEntry 2 }

ieee8021QBridgeStaticMulticastStaticEgressPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The set of ports to which frames received from a
specific port and destined for a specific Multicast or
Broadcast MAC address must be forwarded, regardless of
any dynamic information, e.g., from MMRP. A port may not
be added in this set if it is already a member of the
set of ports in ieee8021QBridgeStaticMulticastForbiddenEgressPorts.
The default value of this object is a string of ones of
appropriate length."
DEFVAL      { ''H }
 ::= { ieee8021QBridgeStaticMulticastEntry 3 }

ieee8021QBridgeStaticMulticastForbiddenEgressPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The set of ports to which frames received from a
specific port and destined for a specific Multicast or
Broadcast MAC address must not be forwarded, regardless
of any dynamic information, e.g., from MMRP. A port may
not be added in this set if it is already a member of the
set of ports in ieee8021QBridgeStaticMulticastStaticEgressPorts.
The default value of this object is a string of zeros of
appropriate length."
DEFVAL      { ''H }
 ::= { ieee8021QBridgeStaticMulticastEntry 4 }

ieee8021QBridgeStaticMulticastStorageType OBJECT-TYPE
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The storage type for this conceptual row. If this object
has a value of permanent(4), then no other objects are
required to be able to be modified."
DEFVAL      { nonVolatile }
 ::= { ieee8021QBridgeStaticMulticastEntry 5 }

ieee8021QBridgeStaticMulticastRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"This object indicates the status of this entry, and is used
to create/delete entries in the table.

An entry of this table may be set to active without setting
any other columns of the table. Also, other columns of this
table may be set while the value of this object is active(1)."
```

```

 ::= { ieee8021QBridgeStaticMulticastEntry 6 }

-- =====
-- The Current VLAN Database
-- =====

ieee8021QBridgeVlanNumDeletes OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "vlan deletions"
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "The number of times a VLAN entry has been deleted from
        the ieee8021QBridgeVlanCurrentTable (for any reason).
        If an entry is deleted, then inserted, and then deleted,
        this counter will be incremented by 2. Discontinuities
        in this value can only occur at a reboot."
 ::= { ieee8021QBridgeVlan 1 }

ieee8021QBridgeVlanCurrentTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeVlanCurrentEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "A table containing current configuration information
        for each VLAN currently configured into the device by
        (local or network) management, or dynamically created
        as a result of MVRP requests received."
    REFERENCE   "12.10.2"
 ::= { ieee8021QBridgeVlan 2 }

ieee8021QBridgeVlanCurrentEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeVlanCurrentEntry
    MAX-ACCESS not-accessible
    STATUS     current
    DESCRIPTION
        "Information for a VLAN configured into the device by
        (local or network) management, or dynamically created
        as a result of MVRP requests received."
    INDEX    { ieee8021QBridgeVlanTimeMark,
              ieee8021QBridgeVlanCurrentComponentId,
              ieee8021QBridgeVlanIndex }
 ::= { ieee8021QBridgeVlanCurrentTable 1 }

Ieee8021QBridgeVlanCurrentEntry :=
SEQUENCE {
    ieee8021QBridgeVlanTimeMark
        TimeFilter,
    ieee8021QBridgeVlanCurrentComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeVlanIndex
        IEEE8021VlanIndex,
    ieee8021QBridgeVlanFdbId
        Unsigned32,
    ieee8021QBridgeVlanCurrentEgressPorts
        PortList,
    ieee8021QBridgeVlanCurrentUntaggedPorts
        PortList,
    ieee8021QBridgeVlanStatus
}

```

```
        INTEGER,
ieee8021QBridgeVlanCreationTime
        TimeTicks
}

ieee8021QBridgeVlanTimeMark OBJECT-TYPE
SYNTAX      TimeFilter
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A TimeFilter for this entry. See the TimeFilter
textual convention to see how this works."
 ::= { ieee8021QBridgeVlanCurrentEntry 1 }

ieee8021QBridgeVlanCurrentComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The component identifier is used to distinguish between the
multiple virtual bridge instances within a PBB. In simple
situations where there is only a single component the default
value is 1."
 ::= { ieee8021QBridgeVlanCurrentEntry 2 }

ieee8021QBridgeVlanIndex OBJECT-TYPE
SYNTAX      IEEE8021VlanIndex
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The VLAN-ID or other identifier referring to this VLAN."
 ::= { ieee8021QBridgeVlanCurrentEntry 3 }

ieee8021QBridgeVlanFdbId OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"The Filtering Database used by this VLAN. This is one
of the ieee8021QBridgeFdbId values in the ieee8021QBridgeFdbTable.
This value is allocated automatically by the device whenever
the VLAN is created: either dynamically by MVRP, or by
management, in ieee8021QBridgeVlanStaticTable. Allocation of this
value follows the learning constraints defined for this
VLAN in ieee8021QBridgeLearningConstraintsTable."
 ::= { ieee8021QBridgeVlanCurrentEntry 4 }

ieee8021QBridgeVlanCurrentEgressPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"The set of ports that are transmitting traffic for
this VLAN as either tagged or untagged frames."
REFERENCE   "12.10.2.1"
 ::= { ieee8021QBridgeVlanCurrentEntry 5 }

ieee8021QBridgeVlanCurrentUntaggedPorts OBJECT-TYPE
SYNTAX      PortList
```

```

MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The set of ports that are transmitting traffic for
     this VLAN as untagged frames."
REFERENCE   "12.10.2.1"
 ::= { ieee8021QBridgeVlanCurrentEntry 6 }

ieee8021QBridgeVlanStatus OBJECT-TYPE
SYNTAX      INTEGER {
            other(1),
            permanent(2),
            dynamicMvrp(3)
        }
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "This object indicates the status of this entry.
     other(1) - this entry is currently in use, but the
                 conditions under which it will remain so differ
                 from the following values.
     permanent(2) - this entry, corresponding to an entry
                     in ieee8021QBridgeVlanStaticTable, is currently in use and
                     will remain so after the next reset of the
                     device. The port lists for this entry include
                     ports from the equivalent ieee8021QBridgeVlanStaticTable
                     entry and ports learned dynamically.
     dynamicMvrp(3) - this entry is currently in use
                     and will remain so until removed by MVRP. There
                     is no static entry for this VLAN, and it will be
                     removed when the last port leaves the VLAN."
 ::= { ieee8021QBridgeVlanCurrentEntry 7 }

ieee8021QBridgeVlanCreationTime OBJECT-TYPE
SYNTAX      TimeTicks
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The value of sysUpTime when this VLAN was created."
 ::= { ieee8021QBridgeVlanCurrentEntry 8 }

-- =====
-- The Static VLAN Database
-- =====

ieee8021QBridgeVlanStaticTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021QBridgeVlanStaticEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table containing static configuration information for
     each VLAN configured into the device by (local or
     network) management. All entries are persistent and will
     be restored after the device is reset."
REFERENCE   "12.7.5"
 ::= { ieee8021QBridgeVlan 3 }

ieee8021QBridgeVlanStaticEntry OBJECT-TYPE
SYNTAX      Ieee8021QBridgeVlanStaticEntry

```

```
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "Static information for a VLAN configured into the
     device by (local or network) management."
INDEX      { ieee8021QBridgeVlanStaticComponentId,
              ieee8021QBridgeVlanStaticVlanIndex }
 ::= { ieee8021QBridgeVlanStaticTable 1 }

Ieee8021QBridgeVlanStaticEntry ::==
SEQUENCE {
    ieee8021QBridgeVlanStaticComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeVlanStaticVlanIndex
        IEEE8021VlanIndex,
    ieee8021QBridgeVlanStaticName
        SnmpAdminString,
    ieee8021QBridgeVlanStaticEgressPorts
        PortList,
    ieee8021QBridgeVlanForbiddenEgressPorts
        PortList,
    ieee8021QBridgeVlanStaticUntaggedPorts
        PortList,
    ieee8021QBridgeVlanStaticRowStatus
        RowStatus
}
ieee8021QBridgeVlanStaticComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The component identifier is used to distinguish between the
     multiple virtual bridge instances within a PBB. In simple
     situations where there is only a single component the default
     value is 1."
 ::= { ieee8021QBridgeVlanStaticEntry 1 }

ieee8021QBridgeVlanStaticVlanIndex OBJECT-TYPE
SYNTAX      IEEE8021VlanIndex
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The VLAN-ID or other identifier referring to this VLAN."
 ::= { ieee8021QBridgeVlanStaticEntry 2 }

ieee8021QBridgeVlanStaticName OBJECT-TYPE
SYNTAX      SnmpAdminString (SIZE (0..32))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "An administratively assigned string, which may be used
     to identify the VLAN."
REFERENCE   "12.10.2.1"
 ::= { ieee8021QBridgeVlanStaticEntry 3 }

ieee8021QBridgeVlanStaticEgressPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-create
```

```
STATUS      current
DESCRIPTION
"The set of ports that are permanently assigned to the
egress list for this VLAN by management. Changes to a
bit in this object affect the per-port, per-VLAN
Registrar control for Registration Fixed for the
relevant MVRP state machine on each port. A port may
not be added in this set if it is already a member of
the set of ports in ieee8021QBridgeVlanForbiddenEgressPorts. The
default value of this object is a string of zeros of
appropriate length, indicating not fixed."
REFERENCE   "12.7.7.3, 11.2.3.2.3"
 ::= { ieee8021QBridgeVlanStaticEntry 4 }
```

```
ieee8021QBridgeVlanForbiddenEgressPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The set of ports that are prohibited by management
from being included in the egress list for this VLAN.
Changes to this object that cause a port to be included
or excluded affect the per-port, per-VLAN Registrar
control for Registration Forbidden for the relevant MVRP
state machine on each port. A port may not be added in
this set if it is already a member of the set of ports
in ieee8021QBridgeVlanStaticEgressPorts. The default value of
this object is a string of zeros of appropriate length,
excluding all ports from the forbidden set."
REFERENCE   "12.7.7.3, 11.2.3.2.3"
 ::= { ieee8021QBridgeVlanStaticEntry 5 }
```

```
ieee8021QBridgeVlanStaticUntaggedPorts OBJECT-TYPE
SYNTAX      PortList
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The set of ports that should transmit egress packets
for this VLAN as untagged. The default value of this
object for the default VLAN (ieee8021QBridgeVlanIndex = 1) is a string
of appropriate length including all ports. There is no
specified default for other VLANs. If a device agent cannot
support the set of ports being set, then it will reject the
set operation with an error. For example, a
manager might attempt to set more than one VLAN to be untagged
on egress where the device does not support this IEEE 802.1Q
option."
REFERENCE   "12.10.2.1"
 ::= { ieee8021QBridgeVlanStaticEntry 6 }
```

```
ieee8021QBridgeVlanStaticRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"This object indicates the status of this entry, and is used
to create/delete entries. Any object in an entry of this table
may be modified while the value of the corresponding instance
of this object is active(1)."
```

```
 ::= { ieee8021QBridgeVlanStaticEntry 7 }

ieee8021QBridgeNextFreeLocalVlanTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeNextFreeLocalVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains information about the next free VLAN
         value for a statically configured VLAN bridge."
    ::= { ieee8021QBridgeVlan 4 }

ieee8021QBridgeNextFreeLocalVlanEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeNextFreeLocalVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The next free VLAN value for a statically configured VLAN bridge"
    INDEX     { ieee8021QBridgeNextFreeLocalVlanComponentId }
    ::= { ieee8021QBridgeNextFreeLocalVlanTable 1 }

Ieee8021QBridgeNextFreeLocalVlanEntry ::=

SEQUENCE {
    ieee8021QBridgeNextFreeLocalVlanComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeNextFreeLocalVlanIndex
        Unsigned32
}

ieee8021QBridgeNextFreeLocalVlanComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021QBridgeNextFreeLocalVlanEntry 1 }

ieee8021QBridgeNextFreeLocalVlanIndex OBJECT-TYPE
    SYNTAX      Unsigned32 (0|4096..4294967295)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The next available value for ieee8021QBridgeVlanIndex of a local
         VLAN entry in ieee8021QBridgeVlanStaticTable. This will report
         values >=4096 if a new Local VLAN may be created or else
         the value 0 if this is not possible.

A row creation operation in this table for an entry with a local
VlanIndex value may fail if the current value of this object
is not used as the index. Even if the value read is used,
there is no guarantee that it will still be the valid index
when the create operation is attempted; another manager may
have already got in during the intervening time interval.
In this case, ieee8021QBridgeNextFreeLocalVlanIndex should be re-read
and the creation re-tried with the new value.

This value will automatically change when the current value is
```

```

        used to create a new row."
 ::= { ieee8021QBridgeNextFreeLocalVlanEntry 2 }

-- =====
-- The VLAN Port Configuration Table
-- =====

ieee8021QBridgePortVlanTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgePortVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing per-port control and status
         information for VLAN configuration in the device."
    REFERENCE   "12.10.1"
 ::= { ieee8021QBridgeVlan 5 }

ieee8021QBridgePortVlanEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgePortVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information controlling VLAN configuration for a port
         on the device. This is indexed by ieee8021BridgeBasePort."
    AUGMENTS { ieee8021BridgeBasePortEntry }
 ::= { ieee8021QBridgePortVlanTable 1 }

Ieee8021QBridgePortVlanEntry ::=
SEQUENCE {
    ieee8021QBridgePvid
        IEEE8021VlanIndex,
    ieee8021QBridgePortAcceptableFrameTypes
        IEEE8021PortAcceptableFrameTypes,
    ieee8021QBridgePortIngressFiltering
        TruthValue,
    ieee8021QBridgePortMvrpEnabledStatus
        TruthValue,
    ieee8021QBridgePortMvrpFailedRegistrations
        Counter64,
    ieee8021QBridgePortMvrpLastPduOrigin
        MacAddress,
    ieee8021QBridgePortRestrictedVlanRegistration
        TruthValue
}

ieee8021QBridgePvid OBJECT-TYPE
    SYNTAX      IEEE8021VlanIndex
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The PVID, the VLAN-ID assigned to untagged frames or
         Priority-Tagged frames received on this port.

        The value of this object MUST be retained across
         reinitializations of the management system."
    REFERENCE   "12.10.1.1"
    DEFVAL     { 1 }
 ::= { ieee8021QBridgePortVlanEntry 1 }

```

```
ieee8021QBridgePortAcceptableFrameTypes OBJECT-TYPE
    SYNTAX      IEEE8021PortAcceptableFrameTypes
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "When this is admitTagged(3), the device will
         discard untagged frames or Priority-Tagged frames
         received on this port. When admitAll(1), untagged
         frames or Priority-Tagged frames received on this port
         will be accepted and assigned to a VID based on the
         PVID and VID Set for this port."
```

This control does not affect VLAN-independent Bridge Protocol Data Unit (BPDU) frames, such as MVRP and Spanning Tree Protocol (STP). It does affect VLAN-dependent BPDU frames, such as MMRP.

The value of this object MUST be retained across reinitializations of the management system."

```
REFERENCE  "12.10.1.3"
DEFVAL     { admitAll }
 ::= { ieee8021QBridgePortVlanEntry 2 }
```

```
ieee8021QBridgePortIngressFiltering OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "When this is true(1), the device will discard incoming
         frames for VLANs that do not include this Port in its
         Member set. When false(2), the port will accept all
         incoming frames."
```

This control does not affect VLAN-independent BPDU frames, such as MVRP and STP. It does affect VLAN-dependent BPDU frames, such as MMRP.

The value of this object MUST be retained across reinitializations of the management system."

```
REFERENCE  "12.10.1.4"
DEFVAL     { false }
 ::= { ieee8021QBridgePortVlanEntry 3 }
```

```
ieee8021QBridgePortMvrpEnabledStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The state of MVRP operation on this port. The value
         true(1) indicates that MVRP is enabled on this port,
         as long as ieee8021QBridgeMvrpEnabledStatus is also enabled
         for this device. When false(2) but
         ieee8021QBridgeMvrpEnabledStatus is still
         enabled for the device, MVRP is disabled on this port:
         any MVRP packets received will be silently discarded, and
         no MVRP registrations will be propagated from other
         ports. This object affects all MVRP Applicant and
         Registrar state machines on this port. A transition
         from false(2) to true(1) will cause a reset of all"
```

MVRP state machines on this port.

The value of this object MUST be retained across reinitializations of the management system."

```
DEFVAL { true }
 ::= { ieee8021QBridgePortVlanEntry 4 }
```

```
ieee8021QBridgePortMvrpFailedRegistrations OBJECT-TYPE
SYNTAX Counter64
UNITS "failed MVRP registrations"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The total number of failed MVRP registrations, for any reason, on this port.
```

Discontinuities in the value of the counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any)."

```
 ::= { ieee8021QBridgePortVlanEntry 5 }
```

```
ieee8021QBridgePortMvrpLastPduOrigin OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The Source MAC Address of the last MVRP message received on this port."
 ::= { ieee8021QBridgePortVlanEntry 6 }
```

```
ieee8021QBridgePortRestrictedVlanRegistration OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The state of Restricted VLAN Registration on this port. If the value of this control is true(1), then creation of a new dynamic VLAN entry is permitted only if there is a Static VLAN Registration Entry for the VLAN concerned, in which the Registrar Administrative Control value for this port is Normal Registration.
```

The value of this object MUST be retained across reinitializations of the management system."

```
REFERENCE "11.2.3.2.3, 12.10.1.7."
DEFVAL { false }
 ::= { ieee8021QBridgePortVlanEntry 7 }
```

```
-- =====
-- Per port VLAN Statistics Table
-- =====
```

```
ieee8021QBridgePortVlanStatisticsTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021QBridgePortVlanStatisticsEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
```

```
"A table containing per-port, per-VLAN statistics for
traffic received."
 ::= { ieee8021QBridgeVlan 6 }

ieee8021QBridgePortVlanStatisticsEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgePortVlanStatisticsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Traffic statistics for a VLAN on an interface."
    INDEX      { ieee8021BridgeBasePortComponentId,
                  ieee8021BridgeBasePort,
                  ieee8021QBridgeVlanIndex }
    ::= { ieee8021QBridgePortVlanStatisticsTable 1 }

Ieee8021QBridgePortVlanStatisticsEntry ::=*
    SEQUENCE {
        ieee8021QBridgeTpVlanPortInFrames
            Counter64,
        ieee8021QBridgeTpVlanPortOutFrames
            Counter64,
        ieee8021QBridgeTpVlanPortInDiscards
            Counter64
    }

ieee8021QBridgeTpVlanPortInFrames OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of valid frames received by this port from
         its segment that were classified as belonging to this
         VLAN. Note that a frame received on this port is
         counted by this object if and only if it is for a
         protocol being processed by the local forwarding process
         for this VLAN. This object includes received bridge
         management frames classified as belonging to this VLAN
         (e.g., MMRP, but not MVRP or STP).

         Discontinuities in the value of the counter can occur
         at re-initialization of the management system, and at
         other times as indicated by the value of
         ifCounterDiscontinuityTime object of the associated
         interface (if any)."
    REFERENCE   "12.6.1.1.3(a)"
    ::= { ieee8021QBridgePortVlanStatisticsEntry 1 }

ieee8021QBridgeTpVlanPortOutFrames OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of valid frames transmitted by this port to
         its segment from the local forwarding process for this
         VLAN. This includes bridge management frames originated
         by this device that are classified as belonging to this
         VLAN (e.g., MMRP, but not MVRP or STP).
```

```

Discontinuities in the value of the counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime object of the associated
interface (if any)."
REFERENCE    "12.6.1.1.3(d)"
 ::= { ieee8021QBridgePortVlanStatisticsEntry 2 }

ieee8021QBridgeTpVlanPortInDiscards OBJECT-TYPE
SYNTAX      Counter64
UNITS      "frames"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"The number of valid frames received by this port from
its segment that were classified as belonging to this
VLAN and that were discarded due to VLAN-related reasons.
Specifically, the IEEE 802.1Q counters for Discard
Inbound and Discard on Ingress Filtering.

Discontinuities in the value of the counter can occur
at re-initialization of the management system, and at
other times as indicated by the value of
ifCounterDiscontinuityTime object of the associated
interface (if any)."
REFERENCE    "12.6.1.1.3"
 ::= { ieee8021QBridgePortVlanStatisticsEntry 3 }

-- =====
-- The VLAN Learning Constraints Table
-- =====

ieee8021QBridgeLearningConstraintsTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021QBridgeLearningConstraintsEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table containing learning constraints for sets of
Shared and Independent VLANs. Entries in this table are
persistent and are preserved across reboots."
REFERENCE    "12.10.3.1"
 ::= { ieee8021QBridgeVlan 8 }

ieee8021QBridgeLearningConstraintsEntry OBJECT-TYPE
SYNTAX      Ieee8021QBridgeLearningConstraintsEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A learning constraint defined for a VLAN."
INDEX      { ieee8021QBridgeLearningConstraintsComponentId,
            ieee8021QBridgeLearningConstraintsVlan,
            ieee8021QBridgeLearningConstraintsSet }
 ::= { ieee8021QBridgeLearningConstraintsTable 1 }

Ieee8021QBridgeLearningConstraintsEntry ::=

SEQUENCE {
    ieee8021QBridgeLearningConstraintsComponentId
    IEEE8021PbbComponentIdentifier,
}

```

```
ieee8021QBridgeLearningConstraintsVlan
    IEEE8021VlanIndex,
ieee8021QBridgeLearningConstraintsSet
    Integer32,
ieee8021QBridgeLearningConstraintsType
    INTEGER,
ieee8021QBridgeLearningConstraintsStatus
    RowStatus
}

ieee8021QBridgeLearningConstraintsComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The component identifier is used to distinguish between the
multiple virtual bridge instances within a PBB. In simple
situations where there is only a single component the default
value is 1."
 ::= { ieee8021QBridgeLearningConstraintsEntry 1 }

ieee8021QBridgeLearningConstraintsVlan OBJECT-TYPE
SYNTAX      IEEE8021VlanIndex
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The index of the row in ieee8021QBridgeVlanCurrentTable for the
VLAN constrained by this entry."
 ::= { ieee8021QBridgeLearningConstraintsEntry 2 }

ieee8021QBridgeLearningConstraintsSet OBJECT-TYPE
SYNTAX      Integer32 (0..65535)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The identity of the constraint set to which
ieee8021QBridgeLearningConstraintsVlan belongs. These values may
be chosen by the management station."
 ::= { ieee8021QBridgeLearningConstraintsEntry 3 }

ieee8021QBridgeLearningConstraintsType OBJECT-TYPE
SYNTAX      INTEGER {
                independent(1),
                shared(2)
            }
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The type of constraint this entry defines.
independent(1) - the VLAN, ieee8021QBridgeLearningConstraintsVlan,
uses a filtering database independent from all
other VLANs in the same set, defined by
ieee8021QBridgeLearningConstraintsSet.
shared(2) - the VLAN, ieee8021QBridgeLearningConstraintsVlan,
shares the same filtering database as all other VLANs
in the same set, defined by
ieee8021QBridgeLearningConstraintsSet."
 ::= { ieee8021QBridgeLearningConstraintsEntry 4 }
```

```

ieee8021QBridgeLearningConstraintsStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The status of this entry. Any object in an entry of this table
         may be modified while the value of the corresponding instance
         of this object is active(1)."
    ::= { ieee8021QBridgeLearningConstraintsEntry 5 }

ieee8021QBridgeLearningConstraintDefaultsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021QBridgeLearningConstraintDefaultsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing learning constraints for sets of
         Shared and Independent VLANs."
    REFERENCE   "12.10.3.1"
    ::= { ieee8021QBridgeVlan 9 }

Ieee8021QBridgeLearningConstraintDefaultsEntry OBJECT-TYPE
    SYNTAX      Ieee8021QBridgeLearningConstraintDefaultsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A learning constraint defined for a VLAN."
    INDEX      { ieee8021QBridgeLearningConstraintDefaultsComponentId }
    ::= { ieee8021QBridgeLearningConstraintDefaultsTable 1 }

Ieee8021QBridgeLearningConstraintDefaultsEntry ::=

SEQUENCE {
    ieee8021QBridgeLearningConstraintDefaultsComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021QBridgeLearningConstraintDefaultsSet
        Integer32,
    ieee8021QBridgeLearningConstraintDefaultsType
        INTEGER
}

ieee8021QBridgeLearningConstraintDefaultsComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021QBridgeLearningConstraintDefaultsEntry 1 }

ieee8021QBridgeLearningConstraintDefaultsSet OBJECT-TYPE
    SYNTAX      Integer32 (0..65535)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The identity of the constraint set to which a VLAN
         belongs, if there is not an explicit entry for that VLAN
         in ieee8021QBridgeLearningConstraintsTable.

```

```
The value of this object MUST be retained across
reinitializations of the management system."
 ::= { ieee8021QBridgeLearningConstraintDefaultsEntry 2 }

ieee8021QBridgeLearningConstraintDefaultsType OBJECT-TYPE
SYNTAX      INTEGER {
              independent(1),
              shared(2)
            }
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"The type of constraint set to which a VLAN belongs, if
there is not an explicit entry for that VLAN in
ieee8021QBridgeLearningConstraintsTable. The types are as defined
for ieee8021QBridgeLearningConstraintsType.

The value of this object MUST be retained across
reinitializations of the management system."
 ::= { ieee8021QBridgeLearningConstraintDefaultsEntry 3 }

-- =====
-- ieee8021QBridgeProtocol subtree
-- =====

ieee8021QBridgeProtocolGroupTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021QBridgeProtocolGroupEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table that contains mappings from Protocol
Templates to Protocol Group Identifiers used for
Port-and-Protocol-based VLAN Classification.

Entries in this table must be persistent over power
up restart/reboot."
REFERENCE   "12.10.1"
 ::= { ieee8021QBridgeProtocol 1 }

ieee8021QBridgeProtocolGroupEntry OBJECT-TYPE
SYNTAX      Ieee8021QBridgeProtocolGroupEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A mapping from a Protocol Template to a Protocol
Group Identifier."
REFERENCE   "12.10.1.1.3 d)"
INDEX       { ieee8021QBridgeProtocolComponentId,
              ieee8021QBridgeProtocolTemplateFrameType,
              ieee8021QBridgeProtocolTemplateProtocolValue }
 ::= { ieee8021QBridgeProtocolGroupTable 1 }

Ieee8021QBridgeProtocolGroupEntry :=
SEQUENCE {
  ieee8021QBridgeProtocolGroupComponentId
    IEEE8021PbbComponentIdentifier,
  ieee8021QBridgeProtocolTemplateFrameType
    INTEGER,
  ieee8021QBridgeProtocolTemplateProtocolValue
```

```

        OCTET STRING,
ieee8021QBridgeProtocolGroupId
    Integer32,
ieee8021QBridgeProtocolGroupRowStatus
    RowStatus
}

ieee8021QBridgeProtocolGroupComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The component identifier is used to distinguish between the
    multiple virtual bridge instances within a PBB. In simple
    situations where there is only a single component the default
    value is 1."
 ::= { ieee8021QBridgeProtocolGroupEntry 1 }

ieee8021QBridgeProtocolTemplateFrameType OBJECT-TYPE
SYNTAX      INTEGER {
    ethernet  (1),
    rfc1042  (2),
    snap8021H (3),
    snapOther (4),
    llcOther   (5)
}
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The data-link encapsulation format or the
    'detagged_frame_type' in a Protocol Template."
REFERENCE   "12.10.1.8"
 ::= { ieee8021QBridgeProtocolGroupEntry 2 }

ieee8021QBridgeProtocolTemplateProtocolValue OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE (2 | 5))
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The identification of the protocol above the data-link
    layer in a Protocol Template. Depending on the
    frame type, the octet string will have one of the
    following values:

    For 'ethernet', 'rfc1042' and 'snap8021H',
    this is the 16-bit (2-octet) IEEE 802.3 Type Field.
    For 'snapOther',
    this is the 40-bit (5-octet) PID.
    For 'llcOther',
    this is the 2-octet IEEE 802.2 Link Service Access
    Point (LSAP) pair: first octet for Destination Service
    Access Point (DSAP) and second octet for Source Service
    Access Point (SSAP)."
REFERENCE   "12.10.1.8"
 ::= { ieee8021QBridgeProtocolGroupEntry 3 }

ieee8021QBridgeProtocolGroupId OBJECT-TYPE
SYNTAX      Integer32 (0..2147483647)
MAX-ACCESS  read-create

```

```
STATUS      current
DESCRIPTION
    "Represents a group of protocols that are associated
     together when assigning a VID to a frame."
REFERENCE   "12.10.1.8"
 ::= { ieee8021QBridgeProtocolGroupEntry 4 }

ieee8021QBridgeProtocolGroupRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object indicates the status of this entry."
 ::= { ieee8021QBridgeProtocolGroupEntry 5 }

ieee8021QBridgeProtocolPortTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021QBridgeProtocolPortEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table that contains VID sets used for
     Port-and-Protocol-based VLAN Classification."
REFERENCE   "12.10.1"
 ::= { ieee8021QBridgeProtocol 2 }

Ieee8021QBridgeProtocolPortEntry OBJECT-TYPE
SYNTAX      Ieee8021QBridgeProtocolPortEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A VID set for a port."
REFERENCE   "12.10.1.1.3 c)"
INDEX       { ieee8021BridgeBasePortComponentId,
              ieee8021BridgeBasePort,
              ieee8021QBridgeProtocolPortGroupId }
 ::= { ieee8021QBridgeProtocolPortTable 1 }

Ieee8021QBridgeProtocolPortEntry ::=
SEQUENCE {
    ieee8021QBridgeProtocolPortGroupId
        Integer32,
    ieee8021QBridgeProtocolPortGroupVid
        VlanId,
    ieee8021QBridgeProtocolPortRowStatus
        RowStatus
}

ieee8021QBridgeProtocolPortGroupId OBJECT-TYPE
SYNTAX      Integer32 (1..2147483647)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "Designates a group of protocols in the Protocol
     Group Database."
REFERENCE   "12.10.1.2"
 ::= { ieee8021QBridgeProtocolPortEntry 1 }

ieee8021QBridgeProtocolPortGroupVid OBJECT-TYPE
SYNTAX      VlanId
```

```
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The VID associated with a group of protocols for
     each port."
REFERENCE   "12.10.1.2"
 ::= { ieee8021QBridgeProtocolPortEntry 2 }

ieee8021QBridgeProtocolPortRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object indicates the status of this entry."
 ::= { ieee8021QBridgeProtocolPortEntry 3 }

-- =====
-- IEEE 802.1Q MIB - Conformance Information
-- =====

ieee8021QBridgeConformance
OBJECT IDENTIFIER ::= { ieee8021QBridgeMib 2 }

ieee8021QBridgeGroups
OBJECT IDENTIFIER ::= { ieee8021QBridgeConformance 1 }

ieee8021QBridgeCompliances
OBJECT IDENTIFIER ::= { ieee8021QBridgeConformance 2 }

-- =====
-- units of conformance
-- =====

ieee8021QBridgeBaseGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeVlanVersionNumber,
    ieee8021QBridgeMaxVlanId,
    ieee8021QBridgeMaxSupportedVlans,
    ieee8021QBridgeNumVlans,
    ieee8021QBridgeMvrpEnabledStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects providing device-level control
     and status information for the Virtual LAN bridge
     services."
 ::= { ieee8021QBridgeGroups 1 }

ieee8021QBridgeFdbUnicastGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeFdbDynamicCount,
    ieee8021QBridgeFdbLearnedEntryDiscards,
    ieee8021QBridgeFdbAgingTime,
    ieee8021QBridgeTpFdbPort,
    ieee8021QBridgeTpFdbStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects providing information about all
```

```
        unicast addresses, learned dynamically or statically
        configured by management, in each Filtering Database."
 ::= { ieee8021QBridgeGroups 2 }

ieee8021QBridgeFdbMulticastGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeTpGroupEgressPorts,
    ieee8021QBridgeTpGroupLearnt
}
STATUS      current
DESCRIPTION
    "A collection of objects providing information about all
     multicast addresses, learned dynamically or statically
     configured by management, in each Filtering Database."
 ::= { ieee8021QBridgeGroups 3 }

ieee8021QBridgeServiceRequirementsGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeForwardAllPorts,
    ieee8021QBridgeForwardAllStaticPorts,
    ieee8021QBridgeForwardAllForbiddenPorts,
    ieee8021QBridgeForwardUnregisteredPorts,
    ieee8021QBridgeForwardUnregisteredStaticPorts,
    ieee8021QBridgeForwardUnregisteredForbiddenPorts
}
STATUS      current
DESCRIPTION
    "A collection of objects providing information about
     service requirements, learned dynamically or statically
     configured by management, in each Filtering Database."
 ::= { ieee8021QBridgeGroups 4 }

ieee8021QBridgeFdbStaticGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeStaticUnicastStaticEgressPorts,
    ieee8021QBridgeStaticUnicastForbiddenEgressPorts,
    ieee8021QBridgeStaticUnicastStorageType,
    ieee8021QBridgeStaticUnicastRowStatus,
    ieee8021QBridgeStaticMulticastStaticEgressPorts,
    ieee8021QBridgeStaticMulticastForbiddenEgressPorts,
    ieee8021QBridgeStaticMulticastStorageType,
    ieee8021QBridgeStaticMulticastRowStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects providing information about
     unicast and multicast addresses statically configured by
     management, in each Filtering Database or VLAN."
 ::= { ieee8021QBridgeGroups 5 }

ieee8021QBridgeVlanGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeVlanNumDeletes,
    ieee8021QBridgeVlanFdbId,
    ieee8021QBridgeVlanCurrentEgressPorts,
    ieee8021QBridgeVlanCurrentUntaggedPorts,
    ieee8021QBridgeVlanStatus,
    ieee8021QBridgeVlanCreationTime
}
```

```
STATUS      current
DESCRIPTION
    "A collection of objects providing information about
    all VLANs currently configured on this device."
::= { ieee8021QBridgeGroups 6 }

ieee8021QBridgeVlanStaticGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeVlanStaticName,
    ieee8021QBridgeVlanStaticEgressPorts,
    ieee8021QBridgeVlanForbiddenEgressPorts,
    ieee8021QBridgeVlanStaticUntaggedPorts,
    ieee8021QBridgeVlanStaticRowStatus,
    ieee8021QBridgeNextFreeLocalVlanIndex
}
STATUS      current
DESCRIPTION
    "A collection of objects providing information about
    VLANs statically configured by management."
::= { ieee8021QBridgeGroups 7 }

ieee8021QBridgeVlanStatisticsGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeTpVlanPortInFrames,
    ieee8021QBridgeTpVlanPortOutFrames,
    ieee8021QBridgeTpVlanPortInDiscards
}
STATUS      current
DESCRIPTION
    "A collection of objects providing per-port packet
    statistics for all VLANs currently configured on this
    device."
::= { ieee8021QBridgeGroups 8 }

ieee8021QBridgeLearningConstraintsGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeLearningConstraintsType,
    ieee8021QBridgeLearningConstraintsStatus
}
STATUS      current
DESCRIPTION
    "A collection of objects defining the Filtering Database
    constraints all VLANs have with each other."
::= { ieee8021QBridgeGroups 9 }

ieee8021QBridgeLearningConstraintDefaultGroup OBJECT-GROUP
OBJECTS {
    ieee8021QBridgeLearningConstraintDefaultsSet,
    ieee8021QBridgeLearningConstraintDefaultsType
}
STATUS      current
DESCRIPTION
    "A collection of objects defining the default Filtering
    Database constraints for VLANs that have no specific
    constraints defined."
::= { ieee8021QBridgeGroups 10 }

ieee8021QBridgeClassificationDeviceGroup OBJECT-GROUP
OBJECTS {
```

```
        ieee8021QBridgeProtocolGroupId,
        ieee8021QBridgeProtocolGroupRowStatus
    }
    STATUS      current
    DESCRIPTION
        "VLAN classification information for the bridge."
    ::= { ieee8021QBridgeGroups 11 }

ieee8021QBridgeClassificationPortGroup OBJECT-GROUP
    OBJECTS {
        ieee8021QBridgeProtocolPortGroupVid,
        ieee8021QBridgeProtocolPortRowStatus
    }
    STATUS      current
    DESCRIPTION
        "VLAN classification information for individual ports."
    ::= { ieee8021QBridgeGroups 12 }

ieee8021QBridgePortGroup2 OBJECT-GROUP
    OBJECTS {
        ieee8021QBridgePvid,
        ieee8021QBridgePortAcceptableFrameTypes,
        ieee8021QBridgePortIngressFiltering,
        ieee8021QBridgePortMvrpEnabledStatus,
        ieee8021QBridgePortMvrpFailedRegistrations,
        ieee8021QBridgePortMvrpLastPduOrigin,
        ieee8021QBridgePortRestrictedVlanRegistration
    }
    STATUS      current
    DESCRIPTION
        "A collection of objects providing port-level VLAN
        control and status information for all ports."
    ::= { ieee8021QBridgeGroups 13 }

ieee8021QBridgeCVlanPortGroup OBJECT-GROUP
    OBJECTS {
        ieee8021QBridgeCVlanPortRowStatus
    }
    STATUS      current
    DESCRIPTION
        "Objects used to create/delete customer VLAN ports."
    ::= { ieee8021QBridgeGroups 14 }

-- =====
-- compliance statements
-- =====

ieee8021QBridgeCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for device support of Virtual
        LAN Bridge services."

    MODULE
        MANDATORY-GROUPS {
            ieee8021QBridgeBaseGroup,
            ieee8021QBridgeVlanGroup,
            ieee8021QBridgeVlanStaticGroup,
            ieee8021QBridgePortGroup2
```

```
}
```

GROUP ieee8021QBridgeFdbUnicastGroup
DESCRIPTION
 "This group is mandatory for bridges that implement
 802.1Q transparent bridging."

GROUP ieee8021QBridgeFdbMulticastGroup
DESCRIPTION
 "This group is mandatory for bridges that implement
 802.1Q transparent bridging."

GROUP ieee8021QBridgeServiceRequirementsGroup
DESCRIPTION
 "This group is mandatory for bridges that implement
 extended filtering services. All objects must be
 read-write if extended-filtering services are
 enabled."

GROUP ieee8021QBridgeFdbStaticGroup
DESCRIPTION
 "This group is optional."

GROUP ieee8021QBridgeVlanStatisticsGroup
DESCRIPTION
 "This group is optional as there may be significant
 implementation cost associated with its support."

GROUP ieee8021QBridgeLearningConstraintsGroup
DESCRIPTION
 "This group is mandatory for devices implementing
 both Independent VLAN Learning (IVL) and Shared
 VLAN Learning (SVL) modes of operation of the
 filtering database, as defined by IEEE 802.1Q."

GROUP ieee8021QBridgeLearningConstraintDefaultGroup
DESCRIPTION
 "This group is mandatory for devices implementing
 both Independent VLAN Learning (IVL) and Shared
 VLAN Learning (SVL) modes of operation of the
 filtering database, as defined by IEEE 802.1Q."

GROUP ieee8021QBridgeClassificationDeviceGroup
DESCRIPTION
 "This group is mandatory ONLY for devices implementing
 VLAN Classification as specified in IEEE 802.1v."

GROUP ieee8021QBridgeClassificationPortGroup
DESCRIPTION
 "This group is mandatory ONLY for devices implementing
 VLAN Classification as specified in IEEE 802.1v."

GROUP ieee8021QBridgeCVlanPortGroup
DESCRIPTION
 "This group is mandatory ONLY for devices supporting
 creation/deletion of customer VLAN ports."

OBJECT ieee8021QBridgePortAcceptableFrameTypes
MIN-ACCESS read-only
DESCRIPTION

"Write access is not required as this is an optional capability in IEEE 802.1Q."

OBJECT ieee8021QBridgePortIngressFiltering
MIN-ACCESS read-only
DESCRIPTION
"Write access is not required as this is an optional capability in IEEE 802.1Q."

OBJECT ieee8021QBridgeLearningConstraintDefaultsSet
MIN-ACCESS read-only
DESCRIPTION
"Write access is not required as this is an optional capability in IEEE 802.1Q."

OBJECT ieee8021QBridgeLearningConstraintDefaultsType
MIN-ACCESS read-only
DESCRIPTION
"Write access is not required as this is an optional capability in IEEE 802.1Q."

OBJECT ieee8021QBridgeProtocolGroupId
MIN-ACCESS read-only
DESCRIPTION
"Write access is not required as this is an optional capability in IEEE 802.1v."

OBJECT ieee8021QBridgeProtocolGroupRowStatus
MIN-ACCESS read-only
DESCRIPTION
"Write access is not required as this is an optional capability in IEEE 802.1v."

::= { ieee8021QBridgeCompliances 1 }

END

17.7.5 Definitions for the IEEE8021-PB MIB module

```

IEEE8021-PB-MIB DEFINITIONS ::= BEGIN

IMPORTS
  MODULE-IDENTITY, OBJECT-TYPE
    FROM SNMPv2-SMI
  TruthValue, RowStatus
    FROM SNMPv2-TC
  ieee802dot1mibs, IEEE8021PbbComponentIdentifierOrZero,
  IEEE8021PbbComponentIdentifier, IEEE8021BridgePortNumber,
  IEEE8021PortAcceptableFrameTypes, IEEE8021PriorityValue,
  IEEE8021BridgePortNumberOrZero
    FROM IEEE8021-TC-MIB
  ieee8021BridgeBasePortComponentId, ieee8021BridgeBasePort
    FROM IEEE8021-BRIDGE-MIB
  VlanId, VlanIdOrNone
    FROM Q-BRIDGE-MIB
  MODULE-COMPLIANCE, OBJECT-GROUP
    FROM SNMPv2-CONF;

ieee8021PbMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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 Levi
  Postal: C/O IEEE 802.1 Working Group
  IEEE Standards Association
  445 Hoes Lane
  P.O. Box 1331
  Piscataway
  NJ 08855-1331
  USA
  E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
  "Provider Bridge MIB module.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices.

REVISION      "201102270000Z" -- February 27, 2011
DESCRIPTION
  "Change to ieee8021PbEdgePortAcceptableFrameTypes
  permissible values, addition of
  IEEE8021BridgePortNumberOrZero to IMPORTS,
  as part of 2011 revision of IEEE Std 802.1Q."
REVISION      "201008260000Z" -- August 26, 2010
DESCRIPTION
  "Minor edits to contact information etc. as part of

```

revision of Std 802.1Q."

REVISION "200810150000Z" -- October 15, 2008
DESCRIPTION
"Initial version."
 ::= { ieee802dot1mibs 5 }

ieee8021PbNotifications OBJECT IDENTIFIER ::= { ieee8021PbMib 0 }
ieee8021PbObjects OBJECT IDENTIFIER ::= { ieee8021PbMib 1 }
ieee8021PbConformance OBJECT IDENTIFIER ::= { ieee8021PbMib 2 }

-- =====
-- ieee8021PbVidTranslationTable:
-- =====

ieee8021PbVidTranslationTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021PbVidTranslationEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"This table is used to configure the VID Translation Table defined in 12.13.2 a) of 802.1Q-2006. The VID Translation Table is used to implement a bi-directional mapping between a local S-VID, used in data and protocol frames transmitted and received through a CNP or PNP, and a relay S-VID, used by the filtering and forwarding process. Each row in this table is indexed by component, port, and local S-VID value and indicates the relay S-VID value to be used for the specified S-VID. If no entry for a component, port, and local-svid is present in this table is present then the relay S-VID used for a frame received on that port with the local S-VID value will be the S-VID that has the same numeric value as the local S-VID of the received frame.

Entries in this table must be persistent over power up restart/reboot."
REFERENCE "12.13.2 a), 12.13.2.1, 12.13.2.2"
 ::= { ieee8021PbObjects 1 }

ieee8021PbVidTranslationEntry OBJECT-TYPE
SYNTAX Ieee8021PbVidTranslationEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"An entry for the S-VID translation table which includes both the Local and Relay S-VIDs between which the PNP or CNP translates.

Note that the component ID of entries in this table must refer to the S-VLAN Component of a Provider Bridge."
INDEX { ieee8021BridgeBasePortComponentId,
ieee8021BridgeBasePort,
ieee8021PbVidTranslationLocalVid }
 ::= { ieee8021PbVidTranslationTable 1 }

Ieee8021PbVidTranslationEntry ::= SEQUENCE {
ieee8021PbVidTranslationLocalVid
VlanId,

```

ieee8021PbVidTranslationRelayVid
    VlanId,
ieee8021PbVidTranslationRowStatus
    RowStatus
}

ieee8021PbVidTranslationLocalVid OBJECT-TYPE
SYNTAX      VlanId
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The S-VID on received(transmitted) at the ISS of a CNP or PNP."
::= { ieee8021PbVidTranslationEntry 1 }

ieee8021PbVidTranslationRelayVid OBJECT-TYPE
SYNTAX      VlanId
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The translated S-VID delivered(received) over the EISS from a
    CNP or PNP. The default value of this object on creation will
    be the value of the corresponding instance of
    ieee8021PbVidTranslationLocalVid."
::= { ieee8021PbVidTranslationEntry 2 }

ieee8021PbVidTranslationRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This indicates the status of an entry in this table, and is
    used to create/delete entries.

    It is an implementation specific decision as to whether any
    column in this table may be set while the corresponding
    instance of this object is valid(1)."
::= { ieee8021PbVidTranslationEntry 3 }

-- =====
-- ieee8021PbCVidRegistrationTable:
-- =====

ieee8021PbCVidRegistrationTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbCVidRegistrationEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A table used in a CEP to create the mapping between a C-VID
    and a service represented by an S-VID.

    Note that the component ID of entries in this table must refer
    to the S-VLAN component of a Provider Edge Bridge and the Port
    Number must refer to the port number of the Customer Edge Port
    associated with that Provider Edge Bridge.

    Entries in this table must be persistent over power up
    restart/reboot."
REFERENCE   "12.13.3.1, 12.13.3.2"
::= { ieee8021PbObjects 2 }

```

```
ieee8021PbCVidRegistrationEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbCVidRegistrationEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An element of the C-VID registration table accessed by PB
         C-VLAN component, Customer Edge Port bridge port number, and
         C-VID. Each element contains the mapping between a C-VID and
         the S-VID which carries the service and booleans for handling
         untagged frames at the PEP and CEP."
    INDEX { ieee8021BridgeBasePortComponentId,
            ieee8021BridgeBasePort,
            ieee8021PbCVidRegistrationCvid }
    ::= { ieee8021PbCVidRegistrationTable 1 }

Ieee8021PbCVidRegistrationEntry ::= SEQUENCE {
    ieee8021PbCVidRegistrationCvid
        VlanId,
    ieee8021PbCVidRegistrationSvid
        VlanId,
    ieee8021PbCVidRegistrationUntaggedPep
        TruthValue,
    ieee8021PbCVidRegistrationUntaggedCep
        TruthValue,
    ieee8021PbCVidRegistrationRowStatus
        RowStatus
}

ieee8021PbCVidRegistrationCvid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "C-VID of this C-VID registration entry."
    ::= { ieee8021PbCVidRegistrationEntry 1 }

ieee8021PbCVidRegistrationSvid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "S-VID for this C-VID registration entry."
    ::= { ieee8021PbCVidRegistrationEntry 2 }

ieee8021PbCVidRegistrationUntaggedPep OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "A flag indicating if this C-VID should be carried untagged
         at the PEP. A value of true(1) means untagged."
    DEFVAL { true }
    ::= { ieee8021PbCVidRegistrationEntry 3 }

ieee8021PbCVidRegistrationUntaggedCep OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
```

```

DESCRIPTION
    "A flag indicating if this C-VID should be carried untagged
     at the CEP. A value of true(1) means untagged."
DEFVAL { true }
 ::= { ieee8021PbCVidRegistrationEntry 4 }

ieee8021PbCVidRegistrationRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This indicates the status of an entry in this table, and is
     used to create/delete entries.

The value of ieee8021PbCVidRegistrationSVid must be set before
an entry in this table can be made valid.

It is an implementation specific decision as to whether any
column in this table may be set while the corresponding
instance of this object is valid(1)."
 ::= { ieee8021PbCVidRegistrationEntry 5 }

-- =====
-- ieee8021PbEdgePortTable:
-- =====

ieee8021PbEdgePortTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbEdgePortEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A Provider Edge Port (PEP) table which indicate the subset of
     parameters needed for each PEP."
REFERENCE   "12.13.3.3, 12.13.3.4"
 ::= { ieee8021PbObjects 3 }

ieee8021PbEdgePortEntry OBJECT-TYPE
SYNTAX      Ieee8021PbEdgePortEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "An entry in the PEP table indexed by ComponentID and S-VID and
     containing parameters used to configure ingress filtering on
     the PEP, thus affecting traffic transiting from the provider
     network to the customer edge port. The columns allow the
     default C-VID value and default User Priority to be specified
     and PEP's ingress filtering operation to be controlled.

Note that the component ID of entries in this table must refer
to an S-VLAN component of a provider edge bridge and the Bridge
Port number must refer to the port number of a Customer Edge
Port associated with that Provider Edge Bridge."
INDEX { ieee8021BridgeBasePortComponentId,
        ieee8021BridgeBasePort,
        ieee8021PbEdgePortSVid }
 ::= { ieee8021PbEdgePortTable 1 }

Ieee8021PbEdgePortEntry ::= SEQUENCE {
    ieee8021PbEdgePortSVid
}

```

```
        VlanId,
ieee8021PbEdgePortPVID
        VlanId,
ieee8021PbEdgePortDefaultUserPriority
        IEEE8021PriorityValue,
ieee8021PbEdgePortAcceptableFrameTypes
        IEEE8021PortAcceptableFrameTypes,
ieee8021PbEdgePortEnableIngressFiltering
        TruthValue
}

ieee8021PbEdgePortSVid OBJECT-TYPE
SYNTAX      VlanId
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The 12 bit S-VID associated with the PEP."
::= { ieee8021PbEdgePortEntry 1 }

ieee8021PbEdgePortPVID OBJECT-TYPE
SYNTAX      VlanId
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"A 12-bit C-VID to be used for untagged frames received at
the Provider Edge Port."
::= { ieee8021PbEdgePortEntry 2 }

ieee8021PbEdgePortDefaultUserPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
>An integer range 0-7 to be used for untagged frames received
at the Provider Edge Port."
::= { ieee8021PbEdgePortEntry 3 }

ieee8021PbEdgePortAcceptableFrameTypes OBJECT-TYPE
SYNTAX      IEEE8021PortAcceptableFrameTypes
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"the Acceptable Frame Types for frames received at the PEP.
The permissible values for the parameter are:
1) Admit all frames;
2) Admit only Untagged and Priority-Tagged frames;
3) Admit only VLAN-Tagged frames."
DEFVAL { admitAll }
::= { ieee8021PbEdgePortEntry 4 }

ieee8021PbEdgePortEnableIngressFiltering OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"Filtering parameter for frames received at the PEP.
The permissible values for the parameter are:
true(1) Enabled;
false(2) Disabled."
```

```

DEFVAL { true }
 ::= { ieee8021PbEdgePortEntry 5 }

-- =====
-- ieee8021PbServicePriorityRegenerationTable:
-- =====

ieee8021PbServicePriorityRegenerationTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbServicePriorityRegenerationEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The priority regeneration table for this PEP."
    REFERENCE   "12.13.3.5, 12.13.3.6"
    ::= { ieee8021PbObjects 4 }

ieee8021PbServicePriorityRegenerationEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbServicePriorityRegenerationEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An element of the PEP priority regeneration table indexed
         by Component ID, bridge port number, S-VID, and received
         priority. Each element contains the regenerated priority.

Note that the component ID of entries in this table must refer
to the S-VLAN component of a Provider Edge Bridge and the Port
Number must refer to the port number of the Customer Edge Port
associated with that S-VLAN component."
INDEX { ieee8021BridgeBasePortComponentId,
        ieee8021BridgeBasePort,
        ieee8021PbServicePriorityRegenerationSvid,
        ieee8021PbServicePriorityRegenerationReceivedPriority }
 ::= { ieee8021PbServicePriorityRegenerationTable 1 }

Ieee8021PbServicePriorityRegenerationEntry ::= SEQUENCE {
    ieee8021PbServicePriorityRegenerationSvid
        VlanId,
    ieee8021PbServicePriorityRegenerationReceivedPriority
        IEEE8021PriorityValue,
    ieee8021PbServicePriorityRegenerationRegeneratedPriority
        IEEE8021PriorityValue
}

ieee8021PbServicePriorityRegenerationSvid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "S-VID for this regeneration table entry."
    ::= { ieee8021PbServicePriorityRegenerationEntry 1 }

ieee8021PbServicePriorityRegenerationReceivedPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Received priority for this regeneration table entry."
    ::= { ieee8021PbServicePriorityRegenerationEntry 2 }

```

```
ieee8021PbServicePriorityRegenerationRegeneratedPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The regenerated priority contained in this regeneration table
         entry."
    ::= { ieee8021PbServicePriorityRegenerationEntry 3 }

-- =====
-- ieee8021PbCnpTable
-- =====

ieee8021PbCnpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbCnpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table is used for dynamic creation and deletion of
         Customer Network Ports on S-VLAN components or I-components.
         Creation of an entry in this table will implicitly also
         create a corresponding entry in the ieee8021BridgeBasePortTable.

         Entries in this table must be persistent across reinitializations
         of the management system."
    REFERENCE   "12.3.3"
    ::= { ieee8021PbObjects 5 }

ieee8021PbCnpEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbCnpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Represents a dynamically created Customer Network Port."
    INDEX { ieee8021BridgeBasePortComponentId,
            ieee8021BridgeBasePort }
    ::= { ieee8021PbCnpTable 1 }

Ieee8021PbCnpEntry ::= SEQUENCE {
    ieee8021PbCnpCComponentId
        IEEE8021PbbComponentIdentifierOrZero,
    ieee8021PbCnpSVid
        VlanIdOrNone,
    ieee8021PbCnpRowStatus
        RowStatus
}

ieee8021PbCnpCComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifierOrZero
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The component ID of the C-Vlan component if this is an
         internal customer network port. The value must be 0 for
         an external customer network port.

         This value must be consistent with the value of the
         corresponding instance of ieee8021PbCnpSVid."
```

```

    Both must be non-zero, or both must be zero."
 ::= { ieee8021PbCnpEntry 1 }

ieee8021PbCnpSVid OBJECT-TYPE
    SYNTAX      VlanIdOrNone
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The S-VID for service for an internal customer network port.
        For an external port, this value must be 0.

        This value must be consistent with the value of the
        corresponding instance of ieee8021PbCnpCComponentId.
        Both must be non-zero, or both must be zero."
 ::= { ieee8021PbCnpEntry 2 }

ieee8021PbCnpRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object is used for creation/deletion of entries in
        this table.

        All columns in this table must have valid values before
        this object can be set to active(1).

        While the value of this object is active(1), the values
        of other columns in the same entry may not be modified."
 ::= { ieee8021PbCnpEntry 3 }

-- =====
-- ieee8021PbPnpTable
-- =====

ieee8021PbPnpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbPnpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table is used for dynamic creation and deletion of
        Provider Network Ports on S-VLAN components or B-components.
        Creation of an entry in this table will implicitly also
        create a corresponding entry in the ieee8021BridgeBasePortTable.

        Entries in this table must be persistent across reinitializations
        of the management system."
    REFERENCE   "12.13.3"
 ::= { ieee8021PbObjects 6 }

ieee8021PbPnpEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbPnpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Represents a dynamically created Provider Network Port."
    INDEX { ieee8021BridgeBasePortComponentId,
            ieee8021BridgeBasePort }
 ::= { ieee8021PbPnpTable 1 }

```

```
Ieee8021PbPnpEntry ::= SEQUENCE {
    ieee8021PbPnpRowStatus
        RowStatus
}

ieee8021PbPnpRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "This object is used for creation/deletion of entries in
         this table."
    ::= { ieee8021PbPnpEntry 1 }

-- =====
-- ieee8021PbCepTable
-- =====

ieee8021PbCepTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbCepEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table is used to create Customer Edge Ports, CEPs, on a
         provider edge bridge. It is indexed by the ComponentId of the
         PEB's S-VLAN component and by the port number for the CEP. Note that
         the CEP's port number belongs to the set of port numbers
         associated with the PEB's S-VLAN component.

Entries in this table must be persistent across reinitializations
of the management system. However, note that some column values,
as noted below, may change across system reinitializations."
    ::= { ieee8021PbObjects 7 }

ieee8021PbCepEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbCepEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The entry of the ieee8021PbCepTable. Note that the component
         index must refer to the S-VLAN component of a PEB, and that the port
         number for the CEP is allocated from the port number space of
         that S-VLAN component."
    INDEX { ieee8021BridgeBasePortComponentId,
            ieee8021BridgeBasePort }
    ::= { ieee8021PbCepTable 1 }

Ieee8021PbCepEntry :=
    SEQUENCE {
        ieee8021PbCepCComponentId  IEEE8021PbbComponentIdentifierOrZero,
        ieee8021PbCepCepPortNumber IEEE8021BridgePortNumberOrZero,
        ieee8021PbCepRowStatus      RowStatus
    }

ieee8021PbCepCComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifierOrZero
    MAX-ACCESS  read-only
    STATUS      current
```

DESCRIPTION

"This column is an implementation specific column that may be used to map the C component associated with this CEP to other tables within the system, such as the Entity MIB. This column may not be created or modified by management station action. A value of 0 is always legal, and non-zero values will be interpreted in an implementation specific manner. The value of this column may or may not persist across system restarts."

::= { ieee8021PbCepEntry 1 }

ieee8021PbCepCepPortNumber OBJECT-TYPE
SYNTAX IEEE8021BridgePortNumberOrZero
MAX-ACCESS read-only
STATUS current
DESCRIPTION

"This column is an implementation specific column that may be used to map the CEP to other tables within the system, for example the Entity MIB. This column may not be created or modified by management station action. A value of 0 is always legal, and non-zero values will be interpreted in an implementation specific manner. The value of this column may or may not persist across system restarts."

::= { ieee8021PbCepEntry 2 }

ieee8021PbCepRowStatus OBJECT-TYPE
SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"This indicates the status of the entry, and is used to create and delete entries in this table."
 ::= { ieee8021PbCepEntry 3 }

-- =====
-- Conformance Information
-- =====

ieee8021PbGroups
OBJECT IDENTIFIER ::= { ieee8021PbConformance 1 }
ieee8021PbCompliances
OBJECT IDENTIFIER ::= { ieee8021PbConformance 2 }

-- =====
-- Units of conformance
-- =====

ieee8021PbVidTranslationGroup OBJECT-GROUP
OBJECTS {
 ieee8021PbVidTranslationRelayVid,
 ieee8021PbVidTranslationRowStatus
}
STATUS current
DESCRIPTION
"The collection of objects used to represent a PB C-VID/S-VID translation."
 ::= { ieee8021PbGroups 1 }

ieee8021PbCVidRegistrationGroup OBJECT-GROUP

```
OBJECTS {
    ieee8021PbCVidRegistrationSVid,
    ieee8021PbCVidRegistrationUntaggedPep,
    ieee8021PbCVidRegistrationUntaggedCep,
    ieee8021PbCVidRegistrationRowStatus
}
STATUS      current
DESCRIPTION
    "The collection of objects used to represent a CEP translation."
::= { ieee8021PbGroups 2 }

ieee8021PbEdgePortGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbEdgePortPVID,
    ieee8021PbEdgePortDefaultUserPriority,
    ieee8021PbEdgePortAcceptableFrameTypes,
    ieee8021PbEdgePortEnableIngressFiltering
}
STATUS      current
DESCRIPTION
    "The collection of objects user to represent a PEP."
::= { ieee8021PbGroups 3 }

ieee8021PbServicePriorityRegenerationGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbServicePriorityRegenerationRegeneratedPriority
}
STATUS      current
DESCRIPTION
    "A regenerated priority value for a PEP."
::= { ieee8021PbGroups 4 }

ieee8021PbDynamicCnpGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbCnpCComponentId,
    ieee8021PbCnpSVid,
    ieee8021PbCnpRowStatus
}
STATUS      current
DESCRIPTION
    "A set of objects used for dynamic creation and deletion
     of customer network ports."
::= { ieee8021PbGroups 5 }

ieee8021PbDynamicPnpGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbPnpRowStatus
}
STATUS      current
DESCRIPTION
    "A set of objects used for dynamic creation and deletion
     of provider network ports."
::= { ieee8021PbGroups 6 }

ieee8021PbDynamicCepGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbCepCComponentId,
    ieee8021PbCepCepPortNumber,
    ieee8021PbCepRowStatus
```

```
    }

STATUS      current
DESCRIPTION
    "A set of objects used for dynamic creation and deletion
     of customer edge ports."
::= { ieee8021PbGroups 7 }

-- =====
-- Compliance statements
-- =====

ieee8021PbCompliance MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for device support of Provider
     Bridge services."

MODULE
MANDATORY-GROUPS {
    ieee8021PbVidTranslationGroup,
    ieee8021PbCVidRegistrationGroup,
    ieee8021PbEdgePortGroup,
    ieee8021PbServicePriorityRegenerationGroup
}

GROUP      ieee8021PbDynamicCnpGroup
DESCRIPTION
    "This group is optional and supports dynamic creation
     and deletion of customer network ports."

GROUP      ieee8021PbDynamicPnpGroup
DESCRIPTION
    "This group is optional and supports dynamic creation
     and deletion of provider network ports."

GROUP      ieee8021PbDynamicCepGroup
DESCRIPTION
    "This group is optional and supports dynamic creation
     and deletion of customer edge ports."

::= { ieee8021PbCompliances 1 }

END
```

17.7.6 Definitions for the IEEE8021-MSTP MIB module

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

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, Counter64,
    Unsigned32, TimeTicks
        FROM SNMPv2-SMI
    TruthValue, RowStatus
        FROM SNMPv2-TC
    ieee802dot1mibs, IEEE8021PbbComponentIdentifier,
    IEEE8021BridgePortNumber, IEEE8021VlanIndex,
    IEEE8021MstIdentifier
        FROM IEEE8021-TC-MIB
    BridgeId
        FROM BRIDGE-MIB
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF;

ieee8021MstpMib MODULE-IDENTITY
LAST-UPDATED "201103230000Z" -- March 23, 2011
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 Levi
      Postal: C/O IEEE 802.1 Working Group
              IEEE Standards Association
              445 Hoes Lane
              P.O. Box 1331
              Piscataway
              NJ 08855-1331
              USA
      E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
"The Bridge MIB modules for managing devices that support
IEEE 802.1Q multiple spanning tree groups.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices.""

REVISION      "201103230000Z" -- March 23, 2011
DESCRIPTION
    "Minor edits to contact information, correction to range of
     ieee8021MstpCistMaxHops and addition of fragile bridge
     as part of 2011 revision of IEEE Std 802.1Q.""

REVISION      "200810150000Z" -- October 15, 2008
DESCRIPTION
    "Initial version."
```

```

 ::= { ieee802dot1mibs 6 }

ieee8021MstpNotifications OBJECT IDENTIFIER ::= { ieee8021MstpMib 0 }
ieee8021MstpObjects OBJECT IDENTIFIER ::= { ieee8021MstpMib 1 }
ieee8021MstpConformance OBJECT IDENTIFIER ::= { ieee8021MstpMib 2 }

-- =====
-- MSTP CIST Table
-- =====

ieee8021MstpCistTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021MstpCistEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The Common and Internal Spanning Tree (CIST) Table. Each row in
the table represents information regarding a Bridge's Bridge
Protocol Entity for the CIST.

Note that entries will exist in this table only for bridge
components for which the corresponding instance of
ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
has a value of mstp(2).

This table contains objects corresponding to the following items
from 12.8.1.1 and 12.8.1.3 of IEEE 802.1Q-2005, and the
802.1ah amendment. Some of those items are provided in the
IEEE8021-SPANNING-TREE-MIB as noted below.

From 12.8.1.1:
Items a), c), o), p), and q) are defined in this table
The remaining items are covered in the
IEEE8021-SPANNING-TREE-MIB:
b) ieee8021SpanningTreeTimeSinceTopologyChange
c) ieee8021SpanningTreeTopChanges
e) ieee8021SpanningTreeDesignatedRoot
f) ieee8021SpanningTreeRootCost
g) ieee8021SpanningTreeRootPort
h) ieee8021SpanningTreeMaxAge
i) ieee8021SpanningTreeForwardDelay
j) ieee8021SpanningTreeBridgeMaxAge
k) ieee8021SpanningTreeBridgeHelloTime
l) ieee8021SpanningTreeBridgeForwardDelay
m) ieee8021SpanningTreeHoldTime
n) ieee8021SpanningTreeVersion

From 12.8.1.3:
Item g) is defined in this table
The remaining items are covered in the
IEEE8021-SPANNING-TREE-MIB:
a) ieee8021SpanningTreeBridgeMaxAge
b) ieee8021SpanningTreeBridgeHelloTime
c) ieee8021SpanningTreeBridgeForwardDelay
d) ieee8021SpanningTreePriority
e) ieee8021SpanningTreeVersion
f) ieee8021RstpTpExtTxHoldCount"
REFERENCE    "12.8.1.1, 12.8.1.3"
 ::= { ieee8021MstpObjects 1 }

ieee8021MstpCistEntry OBJECT-TYPE

```

```
SYNTAX      Ieee8021MstpCistEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A CIST Table entry."
INDEX { ieee8021MstpCistComponentId }
 ::= { ieee8021MstpCistTable 1 }

Ieee8021MstpCistEntry ::= SEQUENCE {
    ieee8021MstpCistComponentId          IEEE8021PbbComponentIdentifier,
    ieee8021MstpCistBridgeIdentifier      BridgeId,
    ieee8021MstpCistTopologyChange       TruthValue,
    ieee8021MstpCistRegionalRootIdentifier BridgeId,
    ieee8021MstpCistPathCost            Unsigned32,
    ieee8021MstpCistMaxHops             Integer32
}

ieee8021MstpCistComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021MstpCistEntry 1 }

ieee8021MstpCistBridgeIdentifier OBJECT-TYPE
    SYNTAX      BridgeId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Bridge Identifier for the CIST."
    REFERENCE  "9.2.5 of IEEE Std 802.1D-2004"
    ::= { ieee8021MstpCistEntry 2 }

ieee8021MstpCistTopologyChange OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an STP Bridge, the value of the Topology Change parameter
         (14.8.1.1.3, item d of IEEE Std 802.1D, 2004 Edition), or in
         an RSTP or MSTP Bridge, asserted if the tcWhile timer for any
         Port for the CIST is non-zero."
    REFERENCE  "14.8.1.1.3:d of IEEE 802.1D-2004"
    ::= { ieee8021MstpCistEntry 3 }

ieee8021MstpCistRegionalRootIdentifier OBJECT-TYPE
    SYNTAX      BridgeId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the CIST Regional Root Identifier parameter,
         i.e. the Bridge Identifier of the current CIST Regional Root."
    REFERENCE  "13.16.4"
    ::= { ieee8021MstpCistEntry 4 }
```

```

ieee8021MstpCistPathCost OBJECT-TYPE
    SYNTAX      Unsigned32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the CIST Path Cost parameter, i.e. the CIST
         path cost from the transmitting Bridge to the CIST Regional Root.
         The sum (about 20 possible out of the given range) of multiple
         port path costs. Also, if the 'transmitting Bridge' is
         the 'CIST Regional Root', then this value could be zero."
    ::= { ieee8021MstpCistEntry 5 }

ieee8021MstpCistMaxHops OBJECT-TYPE
    SYNTAX      Integer32 (6..40)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the MaxHops parameter.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "13.22.1"
    ::= { ieee8021MstpCistEntry 6 }

-- =====
-- ieee8021MstpTable:
-- =====

ieee8021MstpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021MstpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the MSTP Table. Each row in the Table
         represents information regarding a Bridge's Bridge Protocol
         Entity for the specified Spanning Tree instance.

         Entries in this table MUST be retained across
         reinitializations of the management system.

         Note that entries can be created in this table only for bridge
         components for which the corresponding instance of
         ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
         has a value of mstp(2)."
    REFERENCE   "12.8.1.2, 12.8.1.4, 12.12.3.2, 12.12.1"
    ::= { ieee8021MstpObjects 2 }

ieee8021MstpEntry OBJECT-TYPE
    SYNTAX      Ieee8021MstpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A MSTP Table entry."
    INDEX { ieee8021MstpComponentId, ieee8021MstpId }
    ::= { ieee8021MstpTable 1 }

Ieee8021MstpEntry ::= SEQUENCE {
    ieee8021MstpComponentId           IEEE8021PbbComponentIdentifier,
    ieee8021MstpId                   IEEE8021MstIdentifier,
}

```

```
ieee8021MstpBridgeId          BridgeId,
ieee8021MstpTimeSinceTopologyChange TimeTicks,
ieee8021MstpTopologyChanges    Counter64,
ieee8021MstpTopologyChange     TruthValue,
ieee8021MstpDesignatedRoot    BridgeId,
ieee8021MstpRootPathCost      Integer32,
ieee8021MstpRootPort         IEEE8021BridgePortNumber,
ieee8021MstpBridgePriority    Integer32,
ieee8021MstpVids0             OCTET STRING,
ieee8021MstpVids1             OCTET STRING,
ieee8021MstpVids2             OCTET STRING,
ieee8021MstpVids3             OCTET STRING,
ieee8021MstpRowStatus        RowStatus
}

ieee8021MstpComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The component identifier is used to distinguish between the
multiple virtual bridge instances within a PBB. In simple
situations where there is only a single component the default
value is 1."
 ::= { ieee8021MstpEntry 1 }

ieee8021MstpId OBJECT-TYPE
SYNTAX      IEEE8021MstIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"In an MSTP Bridge, this parameter is the MSTID, i.e. the
identifier of a Spanning Tree (or MST) Instance."
 ::= { ieee8021MstpEntry 2 }

ieee8021MstpBridgeId OBJECT-TYPE
SYNTAX      BridgeId
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"In an MSTP Bridge, the Bridge Identifier for the MSTI."
REFERENCE  "13.23.2"
 ::= { ieee8021MstpEntry 3 }

ieee8021MstpTimeSinceTopologyChange OBJECT-TYPE
SYNTAX      TimeTicks
UNITS       "centi-seconds"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
"In an MSTP Bridge, count in seconds of the time elapsed since
tcWhile (13.21) was last non-zero for any Port for the MSTI."
REFERENCE  "13.21"
 ::= { ieee8021MstpEntry 4 }

ieee8021MstpTopologyChanges OBJECT-TYPE
SYNTAX      Counter64
UNITS       "topology changes"
MAX-ACCESS  read-only
```

```

STATUS      current
DESCRIPTION
    "In an MSTP Bridge, count of the times tcWhile (13.21) has been
     non-zero for any Port for the MSTI since the Bridge was powered
     on or initialized."
REFERENCE   "13.21"
 ::= { ieee8021MstpEntry 5 }

ieee8021MstpTopologyChange OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Topology Change parameter value: true(1)
     if tcWhile is non-zero for any Port for the MSTI."
REFERENCE   "13.21"
 ::= { ieee8021MstpEntry 6 }

ieee8021MstpDesignatedRoot OBJECT-TYPE
SYNTAX      BridgeId
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Designated Root parameter value, i.e. the
     Bridge Identifier of the Root Bridge for the MSTI."
REFERENCE   "13.24.2"
 ::= { ieee8021MstpEntry 7 }

ieee8021MstpRootPathCost OBJECT-TYPE
SYNTAX      Integer32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Root Path Cost parameter value, i.e. the
     path cost from the transmitting Bridge to the Root Bridge for
     the MSTI."
REFERENCE   "13.24.2"
 ::= { ieee8021MstpEntry 8 }

ieee8021MstpRootPort OBJECT-TYPE
SYNTAX      IEEE8021BridgePortNumber
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Root Port parameter value, i.e. the Root
     Port for the MSTI."
REFERENCE   "13.23.5"
 ::= { ieee8021MstpEntry 9 }

ieee8021MstpBridgePriority OBJECT-TYPE
SYNTAX      Integer32 (0..61440)
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Bridge Priority parameter value for the
     MSTI, i.e. the four most significant bits of the Bridge Identifier
     for the MSTI."
REFERENCE   "13.23.2"
 ::= { ieee8021MstpEntry 10 }

```

```
ieee8021MstpVids0 OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(128))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the first 1024 bits of the 4096 bit vector
         indicating which VIDs are assigned to this MSTID. The high order
         bit of the first octet corresponds to the first bit of the vector,
         while the low order bit of the last octet corresponds to the last
         bit of this portion of the vector. A bit that is on (equal to 1)
         indicates that the corresponding VID is assigned to this MSTID."
    ::= { ieee8021MstpEntry 11 }

ieee8021MstpVids1 OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(128))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the second 1024 bits of the 4096 bit vector
         indicating which VIDs are assigned to this MSTID. The high order
         bit of the first octet corresponds to the first bit of this
         portion of the vector, while the low order bit of the last octet
         corresponds to the last bit of this portion of the vector. A bit
         that is on (equal to 1) indicates that the corresponding VID is
         assigned to this MSTID."
    ::= { ieee8021MstpEntry 12 }

ieee8021MstpVids2 OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(128))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the third 1024 bits of the 4096 bit vector
         indicating which VIDs are assigned to this MSTID. The high order
         bit of the first octet corresponds to the first bit of this
         portion of the vector, while the low order bit of the last octet
         corresponds to the last bit of this portion of the vector. A bit
         that is on (equal to 1) indicates that the corresponding VID is
         assigned to this MSTID."
    ::= { ieee8021MstpEntry 13 }

ieee8021MstpVids3 OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(128))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the fourth 1024 bits of the 4096 bit vector
         indicating which VIDs are assigned to this MSTID. The high order
         bit of the first octet corresponds to the first bit of this
         portion of the vector, while the low order bit of the last octet
         corresponds to the last bit of this portion of the vector. A bit
         that is on (equal to 1) indicates that the corresponding VID is
         assigned to this MSTID."
    ::= { ieee8021MstpEntry 14 }

ieee8021MstpRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
```

```

STATUS      current
DESCRIPTION
"The status of the row.

Read SNMPv2-TC (RFC2579) for an
explanation of the possible values this object can take.

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."
 ::= { ieee8021MstpEntry 15 }

-- =====
-- ieee8021MstpCistPortTable:
-- =====

ieee8021MstpCistPortTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021MstpCistPortEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The CIST Port Table. Each row in the Table represents information
regarding a specific Port within the Bridge's Bridge Protocol
Entity, for the CIST.

The values of all writable objects in this table MUST be
retained across reinitializations of the management system.

Note that entries will exist in this table only for bridge
components for which the corresponding instance of
ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
has a value of mstp(2).

This table contains objects corresponding to the following items
from 12.8.2.1, 12.8.2.3, and 12.8.1.5 of IEEE 802.1Q-2005, and the
802.1ah amendment. Some of those items are provided in the
IEEE8021-SPANNING-TREE-MIB as noted below.

From 12.8.2.1:
Items a), d), e), and i) through w) are defined in this table
The remaining items are covered in the
IEEE8021-SPANNING-TREE-MIB:
    b) ieee8021SpanningTreePortState
    c) ieee8021SpanningTreePortPriority
    d) ieee8021SpanningTreePortPathCost32,
    f) ieee8021SpanningTreePortDesignatedCost
    g) ieee8021SpanningTreePortDesignatedBridge
    h) ieee8021SpanningTreePortDesignatedPort

From 12.8.2.3:
Items a), b), and d) through h) are defined in this table
(item a is the index)
The remaining items are covered in the
IEEE8021-SPANNING-TREE-MIB:
    b) ieee8021SpanningTreePortPathCost,
    c) ieee8021SpanningTreePortPriority

From 12.8.2.5:
All items are defined in this table
From 802.1ah 12.8.2.1:
Items u), v), w), and x) are defined in this table

```

From 802.1ah 12.8.2.3:
Items i), j), k), and l) are defined in this table"
REFERENCE "12.8.2.1, 12.8.2.3, 12.8.2.5"
 ::= { ieee8021MstpObjects 3 }

ieee8021MstpCistPortEntry OBJECT-TYPE
SYNTAX Ieee8021MstpCistPortEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"A CIST Port Table entry."
INDEX { ieee8021MstpCistPortComponentId, ieee8021MstpCistPortNum }
 ::= { ieee8021MstpCistPortTable 1 }

Ieee8021MstpCistPortEntry ::= SEQUENCE {
ieee8021MstpCistPortComponentId IEEE8021PbbComponentIdentifier,
ieee8021MstpCistPortNum IEEE8021BridgePortNumber,
ieee8021MstpCistPortUptime TimeTicks,
ieee8021MstpCistPortAdminPathCost Integer32,
ieee8021MstpCistPortDesignatedRoot BridgeId,
ieee8021MstpCistPortTopologyChangeAck TruthValue,
ieee8021MstpCistPortHelloTime Integer32,
ieee8021MstpCistPortAdminEdgePort TruthValue,
ieee8021MstpCistPortOperEdgePort TruthValue,
ieee8021MstpCistPortMacEnabled TruthValue,
ieee8021MstpCistPortMacOperational TruthValue,
ieee8021MstpCistPortRestrictedRole TruthValue,
ieee8021MstpCistPortRestrictedTcn TruthValue,
ieee8021MstpCistPortRole INTEGER,
ieee8021MstpCistPortDisputed TruthValue,
ieee8021MstpCistPortCistRegionalRootId BridgeId,
ieee8021MstpCistPortCistPathCost Unsigned32,
ieee8021MstpCistPortProtocolMigration TruthValue,
ieee8021MstpCistPortEnableBPDURx TruthValue,
ieee8021MstpCistPortEnableBPDUTx TruthValue,
ieee8021MstpCistPortPseudoRootId BridgeId,
ieee8021MstpCistPortIsL2Gp TruthValue
}
ieee8021MstpCistPortComponentId OBJECT-TYPE
SYNTAX IEEE8021PbbComponentIdentifier
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The component identifier is used to distinguish between the
multiple virtual bridge instances within a PBB. In simple
situations where there is only a single component the default
value is 1."
 ::= { ieee8021MstpCistPortEntry 1 }

ieee8021MstpCistPortNum OBJECT-TYPE
SYNTAX IEEE8021BridgePortNumber
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The Port's Port Number parameter value for the CIST, i.e. the
number of the Bridge Port for the CIST."
 ::= { ieee8021MstpCistPortEntry 2 }

```

ieee8021MstpCistPortUptime OBJECT-TYPE
    SYNTAX      TimeTicks
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Port's Uptime parameter value for the CIST, i.e. the count
         in seconds of the time elapsed since the Port was last reset or
         initialized (BEGIN, 13.23)."
    ::= { ieee8021MstpCistPortEntry 3 }

ieee8021MstpCistPortAdminPathCost OBJECT-TYPE
    SYNTAX      Integer32 (0..200000000)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administratively assigned value for the contribution
         of this port to the path cost of paths toward the spanning
         tree root.

        Writing a value of '0' assigns the automatically calculated
        default Path Cost value to the port. If the default Path
        Cost is being used, this object returns '0' when read.

        This complements the object ieee8021MstpCistPortPathCost,
        which returns the operational value of the path cost.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "13.22:p, 17.13.11 of IEEE Std 802.1D"
    ::= { ieee8021MstpCistPortEntry 4 }

ieee8021MstpCistPortDesignatedRoot OBJECT-TYPE
    SYNTAX      BridgeId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The CIST Regional Root Identifier component of the Port's port
         priority vector, as defined in 13.10, for the CIST."
    REFERENCE   "13.24.12"
    ::= { ieee8021MstpCistPortEntry 5 }

ieee8021MstpCistPortTopologyChangeAck OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Port's Topology Change Acknowledge parameter value.
         True(1) if a Configuration Message with a topology change
         acknowledge flag set is to be transmitted. "
    REFERENCE   "17.19.41 of IEEE Std 802.1D"
    ::= { ieee8021MstpCistPortEntry 6 }

ieee8021MstpCistPortHelloTime OBJECT-TYPE
    SYNTAX      Integer32 (100..1000)
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION

```

```
"The Port's Hello Time timer parameter value, for the CIST.  
In centi-seconds"  
REFERENCE "13.24.13, 17.19.22 of IEEE Std 802.1D"  
 ::= { ieee8021MstpCistPortEntry 7 }  
  
ieee8021MstpCistPortAdminEdgePort OBJECT-TYPE  
SYNTAX TruthValue  
MAX-ACCESS read-write  
STATUS current  
DESCRIPTION  
"In a Bridge that supports the identification of edge ports, the  
Port's Admin Edge Port parameter value, for the CIST."  
REFERENCE "17.13.1 of IEEE Std 802.1D"  
DEFVAL { true }  
 ::= { ieee8021MstpCistPortEntry 8 }  
  
ieee8021MstpCistPortOperEdgePort OBJECT-TYPE  
SYNTAX TruthValue  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"In a Bridge that supports the identification of edge ports, the  
Port's operational Edge Port parameter value, for the CIST.  
True(1) if it is an Oper Edge Port."  
REFERENCE "17.19.17 of IEEE Std 802.1D"  
 ::= { ieee8021MstpCistPortEntry 9 }  
  
ieee8021MstpCistPortMacEnabled OBJECT-TYPE  
SYNTAX TruthValue  
MAX-ACCESS read-write  
STATUS current  
DESCRIPTION  
"In a Bridge that supports the MAC Enabled parameter, the current  
state of the MAC Enabled parameter.  
True(1) indicates that administratively the MAC is set as if it  
was connected to a point-to-point LAN."  
REFERENCE "12.8.2.1.3 m)"  
 ::= { ieee8021MstpCistPortEntry 10 }  
  
ieee8021MstpCistPortMacOperational OBJECT-TYPE  
SYNTAX TruthValue  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"In a Bridge that supports the MAC Operational parameter, the  
current state of the MAC Operational parameter.  
True(1) indicates the MAC is operational."  
REFERENCE "12.8.2.1.3 n)"  
 ::= { ieee8021MstpCistPortEntry 11 }  
  
ieee8021MstpCistPortRestrictedRole OBJECT-TYPE  
SYNTAX TruthValue  
MAX-ACCESS read-write  
STATUS current  
DESCRIPTION  
"The current state of the restrictedRole parameter for the Port.  
True(1) causes the Port not to be selected as Root Port for the  
CIST or any MSTI. "  
REFERENCE "13.25.14"
```

```
::= { ieee8021MstpCistPortEntry 12 }

ieee8021MstpCistPortRestrictedTcn OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The current state of the restrictedTcn parameter for the Port.
         True(1) causes the Port not to propagate topology changes to
         other Ports."
    REFERENCE   "13.25.15"
    ::= { ieee8021MstpCistPortEntry 13 }

ieee8021MstpCistPortRole OBJECT-TYPE
    SYNTAX      INTEGER {
        root(1),
        alternate(2),
        designated(3),
        backup(4)
    }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current Port Role for the Port (i.e., Root, Alternate,
         Designated, or Backup), for the CIST."
    REFERENCE   "12.2.8.1.3 s)"
    ::= { ieee8021MstpCistPortEntry 14 }

ieee8021MstpCistPortDisputed OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current value of the disputed variable for the CIST for
         the Port. A value of true(1) indicates that the disputed
         variable is set. A value of false(2) indicates that the
         agreed variable is cleared."
    REFERENCE   "13.24:u, and 17.19.6 of IEEE Std 802.1D"
    ::= { ieee8021MstpCistPortEntry 15 }

ieee8021MstpCistPortCistRegionalRootId OBJECT-TYPE
    SYNTAX      BridgeId
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the CIST Regional Root Identifier, i.e. the
         Bridge Identifier of the current CIST Regional Root, for the CIST."
    REFERENCE   "13.9:c, 13.10, 13.24.12"
    ::= { ieee8021MstpCistPortEntry 16 }

ieee8021MstpCistPortCistPathCost OBJECT-TYPE
    SYNTAX      Unsigned32 (0..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the CIST Internal Root Path Cost, i.e. the
         CIST path cost from the transmitting Bridge to the CIST Regional
         Root, for the CIST."
```

The sum (about 20 possible out of the given range) of multiple port path costs. Also, if the 'the transmitting Bridge' is 'the CIST Regional Root', then this value could be zero."

REFERENCE "13.9:d, 13.10, 13.24.12"
 ::= { ieee8021MstpCistPortEntry 17 }

ieee8021MstpCistPortProtocolMigration OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "In an MSTP Bridge, the current value of the mcheck variable for the Port. A value of true(1) forces the state machine to perform functions as per 17.19.13."
REFERENCE "17.19.13 of IEEE Std 802.1D"
 ::= { ieee8021MstpCistPortEntry 18 }

ieee8021MstpCistPortEnableBPDURx OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "In an MSTP Bridge, the enableBPDURx parameter value. A value of false(2) indicates that BPDUs are ignored."
REFERENCE "13.25.18"
DEFVAL { false }
 ::= { ieee8021MstpCistPortEntry 19 }

ieee8021MstpCistPortEnableBPDUtx OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "In an MSTP Bridge, the enableBPDUtx parameter value. A value of false(2) indicates that BPDUs are not transmitted."
REFERENCE "13.25.19"
DEFVAL { false }
 ::= { ieee8021MstpCistPortEntry 20 }

ieee8021MstpCistPortPseudoRootId OBJECT-TYPE
 SYNTAX BridgeId
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "In an MSTP Bridge, the pseudoRootId parameter value."
REFERENCE "13.25.20"
 ::= { ieee8021MstpCistPortEntry 21 }

ieee8021MstpCistPortIsL2Gp OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "In an MSTP Bridge, the isL2gp parameter value. A value of true(1) indicates this is an L2GP port."
REFERENCE "13.25.21"
DEFVAL { false }
 ::= { ieee8021MstpCistPortEntry 22 }

```
-- =====
-- ieee8021MstpPortTable:
-- =====

ieee8021MstpPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021MstpPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The MSTP Port Table. Each row in the Table represents information
         regarding a specific Port within the Bridge's Bridge Protocol
         Entity, for a given MSTI.

The values of all writable objects in this table MUST be
retained across reinitializations of the management system.

Note that entries will exist in this table only for bridge
components for which the corresponding instance of
ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
has a value of mstp(2)."
REFERENCE    "12.8.2.2, 12.8.2.4"
::= { ieee8021MstpObjects 4 }

ieee8021MstpPortEntry OBJECT-TYPE
    SYNTAX      Ieee8021MstpPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A MSTP Port Table entry."
INDEX { ieee8021MstpPortComponentId,
        ieee8021MstpPortMstId,
        ieee8021MstpPortNum }
 ::= { ieee8021MstpPortTable 1 }

Ieee8021MstpPortEntry ::= SEQUENCE {
    ieee8021MstpPortComponentId      IEEE8021PbbComponentIdentifier,
    ieee8021MstpPortMstId           IEEE8021MstIdentifier,
    ieee8021MstpPortNum             IEEE8021BridgePortNumber,
    ieee8021MstpPortUptime          TimeTicks,
    ieee8021MstpPortState           INTEGER,
    ieee8021MstpPortPriority        Integer32,
    ieee8021MstpPortPathCost        Integer32,
    ieee8021MstpPortDesignatedRoot  BridgeId,
    ieee8021MstpPortDesignatedCost  Integer32,
    ieee8021MstpPortDesignatedBridge BridgeId,
    ieee8021MstpPortDesignatedPort  IEEE8021BridgePortNumber,
    ieee8021MstpPortRole            INTEGER,
    ieee8021MstpPortDisputed       TruthValue
}

ieee8021MstpPortComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
```

```
 ::= { ieee8021MstpPortEntry 1 }

ieee8021MstpPortMstId OBJECT-TYPE
    SYNTAX      IEEE8021MstIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, this parameter is the MSTID, i.e. the
         identifier of a Spanning Tree (or MST) Instance."
    ::= { ieee8021MstpPortEntry 2 }

ieee8021MstpPortNum OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the Port's Port Number parameter value for
         the MSTI, i.e. the number of the Bridge Port for the MSTI."
    ::= { ieee8021MstpPortEntry 3 }

ieee8021MstpPortUptime OBJECT-TYPE
    SYNTAX      TimeTicks
    UNITS      "centi-seconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the Port's Uptime parameter value for the
         MSTI, i.e. the count in seconds of the time elapsed since the
         Port was last reset or initialized (BEGIN, 13.23)."
    ::= { ieee8021MstpPortEntry 4 }

ieee8021MstpPortState OBJECT-TYPE
    SYNTAX      INTEGER {
                    disabled(1),
                    listening(2),
                    learning(3),
                    forwarding(4),
                    blocking(5)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the current state of the Port (i.e., Disabled,
         Listening, Learning, Forwarding, or Blocking), for the MSTI."
    REFERENCE   "13.35, and 17.10 of IEEE Std 802.1D"
    ::= { ieee8021MstpPortEntry 5 }

ieee8021MstpPortPriority OBJECT-TYPE
    SYNTAX      Integer32 (0..240)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the Port's Port Priority parameter value for
         the MSTI, i.e. the priority field for the Port Identifier for the
         Port for the MSTI."
    REFERENCE   "13.24.12"
    ::= { ieee8021MstpPortEntry 6 }

ieee8021MstpPortPathCost OBJECT-TYPE
```

```

SYNTAX      Integer32 (1..200000000)
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Port's Port Path Cost parameter value for
     the MSTI."
REFERENCE   "13.37.1"
 ::= { ieee8021MstpPortEntry 7 }

ieee8021MstpPortDesignatedRoot OBJECT-TYPE
SYNTAX      BridgeId
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Regional Root Identifier component of the
     Port's MSTI port priority vector, as defined in 13.11, for the MSTI."
REFERENCE   "13.24.12"
 ::= { ieee8021MstpPortEntry 8 }

ieee8021MstpPortDesignatedCost OBJECT-TYPE
SYNTAX      Integer32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Internal Root Path Cost component of the
     Port's MSTI port priority vector, as defined in 13.11, for the MSTI."
REFERENCE   "13.24.12"
 ::= { ieee8021MstpPortEntry 9 }

ieee8021MstpPortDesignatedBridge OBJECT-TYPE
SYNTAX      BridgeId
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Designated Bridge Identifier component of
     the Port's MSTI port priority vector, as defined in 13.11, for
     the MSTI."
REFERENCE   "13.24.12"
 ::= { ieee8021MstpPortEntry 10 }

ieee8021MstpPortDesignatedPort OBJECT-TYPE
SYNTAX      IEEE8021BridgePortNumber
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Designated Port Identifier component of the
     Port's MSTI port priority vector, as defined in 13.11, for the MSTI."
REFERENCE   "13.24.12"
 ::= { ieee8021MstpPortEntry 11 }

ieee8021MstpPortRole OBJECT-TYPE
SYNTAX      INTEGER {
              root(1),
              alternate(2),
              designated(3),
              backup(4)
            }
MAX-ACCESS  read-only
STATUS      current

```

```
DESCRIPTION
    "In an MSTP Bridge, the current Port Role for the Port (i.e., Root,
     Alternate, Designated, or Backup), for the MSTI."
 ::= { ieee8021MstpPortEntry 12 }

ieee8021MstpPortDisputed OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the current value of the disputed variable for
     the MSTI for the Port."
REFERENCE   "13.24:u, and 17.19.6 of IEEE Std 802.1D"
 ::= { ieee8021MstpPortEntry 13 }

-- =====
-- ieee8021MstpFidToMstiTable
-- =====

ieee8021MstpFidToMstiTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021MstpFidToMstiEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the fixed-length FID to MSTID Allocation Table
     entry. Each entry in the Table corresponds to a FID, and the value
     of the entry specifies the MSTID of the spanning tree to which the
     set of VLANs supported by that FID are assigned. A value of zero
     in an entry specifies that the set of VLANs supported by that FID
     are assigned to the CST.

The values of all writable objects in this table MUST be
retained across reinitializations of the management system.

Note that entries will exist in this table only for bridge
components for which the corresponding instance of
ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
has a value of mstp(2)."
REFERENCE   "12.12.2"
 ::= { ieee8021MstpObjects 5 }

ieee8021MstpFidToMstiEntry OBJECT-TYPE
SYNTAX      Ieee8021MstpFidToMstiEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, a FID to MSTID Allocation Table entry."
INDEX { ieee8021MstpFidToMstiComponentId, ieee8021MstpFidToMstiFid }
 ::= { ieee8021MstpFidToMstiTable 1 }

Ieee8021MstpFidToMstiEntry ::= SEQUENCE {
    ieee8021MstpFidToMstiComponentId  IEEE8021PbbComponentIdentifier,
    ieee8021MstpFidToMstiFid        Unsigned32,
    ieee8021MstpFidToMstiMstId     IEEE8021MstIdentifier
}

ieee8021MstpFidToMstiComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
```

```

STATUS      current
DESCRIPTION
  "The component identifier is used to distinguish between the
  multiple virtual bridge instances within a PBB. In simple
  situations where there is only a single component the default
  value is 1."
 ::= { ieee8021MstpFidToMstiEntry 1 }

ieee8021MstpFidToMstiFid OBJECT-TYPE
  SYNTAX      Unsigned32 (1..4094)
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "In an MSTP Bridge, the FID of the entry in the FID to MSTID
     Allocation Table."
 ::= { ieee8021MstpFidToMstiEntry 2 }

ieee8021MstpFidToMstiMstId OBJECT-TYPE
  SYNTAX      IEEE8021MstIdentifier
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "In an MSTP Bridge, the MSTID to which the FID (of the entry in
     the FID to MSTID Allocation Table) is to be allocated."
 ::= { ieee8021MstpFidToMstiEntry 3 }

-- =====
-- ieee8021MstpVlanTable
-- =====

ieee8021MstpVlanTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF Ieee8021MstpVlanEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "In an MSTP Bridge, the fixed-length (4096 elements), read-only,
     MST Configuration Table. Its elements are derived from other
     configuration information held by the Bridge; specifically, the
     current state of the VID to FID Allocation Table (8.8.7.1,
     12.10.3), and the FID to MSTID Allocation Table (8.9.3, 12.12.2).
     Hence, changes made to either of these Tables can in turn affect
     the contents of the MST Configuration Table, and also affect the
     value of the digest element of the MST Configuration Identifier.

    The values of all writable objects in this table MUST be
    retained across reinitializations of the management system.

    Note that entries will exist in this table only for bridge
    components for which the corresponding instance of
    ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
    has a value of mstp(2)."
  REFERENCE   "12.12.3.1"
 ::= { ieee8021MstpObjects 6 }

ieee8021MstpVlanEntry OBJECT-TYPE
  SYNTAX      Ieee8021MstpVlanEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION

```

```
"In an MSTP Bridge, a MST Configuration Table entry."
INDEX { ieee8021MstpVlanComponentId, ieee8021MstpVlanId }
 ::= { ieee8021MstpVlanTable 1 }

Ieee8021MstpVlanEntry ::= SEQUENCE {
    ieee8021MstpVlanComponentId  IEEE8021PbbComponentIdentifier,
    ieee8021MstpVlanId          IEEE8021VlanIndex,
    ieee8021MstpVlanMstId       IEEE8021MstIdentifier
}

ieee8021MstpVlanComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The component identifier is used to distinguish between the
         multiple virtual bridge instances within a PBB. In simple
         situations where there is only a single component the default
         value is 1."
    ::= { ieee8021MstpVlanEntry 1 }

ieee8021MstpVlanId OBJECT-TYPE
    SYNTAX      IEEE8021VlanIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the VID of the entry in the MST
         Configuration Table."
    ::= { ieee8021MstpVlanEntry 2 }

ieee8021MstpVlanMstId OBJECT-TYPE
    SYNTAX      IEEE8021MstIdentifier
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "In an MSTP Bridge, the MSTID value corresponding to the VID
         of the entry in the MST Configuration Table."
    ::= { ieee8021MstpVlanEntry 3 }

-- =====
-- MST Configuration Identifier Table
-- =====

ieee8021MstpConfigIdTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021MstpConfigIdEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing the MST Configuration Identifier for each
         virtual bridge. In simple situations where there is only
         a single component, there will only be a single entry in
         this table (i.e., only a single MST Configuration Identifier)."

The values of all writable objects in this table MUST be
retained across reinitializations of the management system.

Note that entries will exist in this table only for bridge
components for which the corresponding instance of
ieee8021SpanningTreeVersion (from the IEEE8021-SPANNING-TREE-MIB)
```

```

        has a value of mstp(2)."
REFERENCE    "12.12.3.3, 12.12.3.4"
 ::= { ieee8021MstpObjects 7 }

ieee8021MstpConfigIdEntry OBJECT-TYPE
SYNTAX      Ieee8021MstpConfigIdEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "An entry containing the MST Configuration Identifier of a bridge."
INDEX { ieee8021MstpConfigIdComponentId }
 ::= { ieee8021MstpConfigIdTable 1 }

Ieee8021MstpConfigIdEntry ::= SEQUENCE {
    ieee8021MstpConfigIdComponentId      IEEE8021PbbComponentIdentifier,
    ieee8021MstpConfigIdFormatSelector   Integer32,
    ieee8021MstpConfigurationName       SnmpAdminString,
    ieee8021MstpRevisionLevel          Unsigned32,
    ieee8021MstpConfigurationDigest    OCTET STRING
}

ieee8021MstpConfigIdComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The component identifier is used to distinguish between the
     multiple virtual bridge instances within a PBB. In simple
     situations where there is only a single component the default
     value is 1."
 ::= { ieee8021MstpConfigIdEntry 1 }

ieee8021MstpConfigIdFormatSelector OBJECT-TYPE
SYNTAX      Integer32 (0..0)
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Configuration Identifier Format Selector
     in use by the Bridge, in the MST Configuration Identifier. This
     has a value of 0 to indicate the format specified in IEEE Std 802.1Q."
REFERENCE    "13.7:1"
 ::= { ieee8021MstpConfigIdEntry 2 }

ieee8021MstpConfigurationName OBJECT-TYPE
SYNTAX      SnmpAdminString (SIZE(32))
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Configuration Name in the MST
     Configuration Identifier."
REFERENCE    "13.7:2"
 ::= { ieee8021MstpConfigIdEntry 3 }

ieee8021MstpRevisionLevel OBJECT-TYPE
SYNTAX      Unsigned32 (0..65535)
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Revision Level in the MST

```

```
        Configuration Identifier."
REFERENCE    "13.7:3"
 ::= { ieee8021MstpConfigIdEntry 4 }

ieee8021MstpConfigurationDigest OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE(16))
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "In an MSTP Bridge, the Configuration Digest in the MST
     Configuration Identifier."
REFERENCE    "13.7:4"
 ::= { ieee8021MstpConfigIdEntry 5 }

-- =====
-- Ieee8021MstpCistPortExtensionTable:
-- =====

ieee8021MstpCistPortExtensionTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021MstpCistPortExtensionEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The CIST Port Extensions Table. Each row in the Table represents
information
    regarding a specific Port within the Bridge's Bridge Protocol
    Entity, for the CIST."
REFERENCE    "12.8.2"
 ::= { ieee8021MstpObjects 8 }

ieee8021MstpCistPortExtensionEntry OBJECT-TYPE
SYNTAX      Ieee8021MstpCistPortExtensionEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A list of additional objects containing information
    maintained by every port about the CIST
    state for that port."
AUGMENTS { ieee8021MstpCistPortEntry}
 ::= { ieee8021MstpCistPortExtensionTable 1 }

Ieee8021MstpCistPortExtensionEntry ::=
SEQUENCE {
    ieee8021MstpCistPortAutoEdgePort
        TruthValue,
    ieee8021MstpCistPortAutoIsolatePort
        TruthValue
}

ieee8021MstpCistPortAutoEdgePort OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The administrative value of the Auto Edge Port parameter.
    A value of true(1) indicates if the bridge detection state
    machine (BDM, 13.31) is to detect other bridges
    attached to the LAN, and set
```

```
ieee8021SpanningTreeRstpPortOperEdgePort automatically.
The default value is true(1)
```

This is optional and provided only by implementations that support the automatic identification of edge ports.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.8.2.1.3")"
`::= { ieee8021MstpCistPortExtensionEntry 1 }`

```
ieee8021MstpCistPortAutoIsolatePort OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The operational value of the Isolate Port parameter.
```

A value of true(1) indicates a Designated Port will transition to discarding if both ieee8021SpanningTreeRstpPortAdminEdgePort and ieee8021SpanningTreeRstpPortAutoEdgePort are FALSE and the other bridge presumed to be attached to the same point-to-point LAN does not transmit periodic BPDUs.

This is optional and provided only by implementations that support the automatic identification of fragile bridges."

REFERENCE "12.8.2.1.3")
`::= { ieee8021MstpCistPortExtensionEntry 2 }`

```
-- =====
-- Conformance Information
-- =====
```

```
ieee8021MstpGroups
OBJECT IDENTIFIER ::= { ieee8021MstpConformance 1 }
ieee8021MstpCompliances
OBJECT IDENTIFIER ::= { ieee8021MstpConformance 2 }
```

```
-- =====
-- Units of conformance
-- =====
```

```
ieee8021MstpCistGroup OBJECT-GROUP
OBJECTS {
    ieee8021MstpCistBridgeIdentifier,
    ieee8021MstpCistTopologyChange,
    ieee8021MstpCistRegionalRootIdentifier,
    ieee8021MstpCistPathCost,
    ieee8021MstpCistMaxHops
}
STATUS      current
DESCRIPTION
    "Objects for the CIST group"
::= { ieee8021MstpGroups 1 }
```

```
ieee8021MstpGroup OBJECT-GROUP
```

```
OBJECTS {
    ieee8021MstpBridgeId,
    ieee8021MstpTimeSinceTopologyChange,
    ieee8021MstpTopologyChanges,
    ieee8021MstpTopologyChange,
    ieee8021MstpDesignatedRoot,
    ieee8021MstpRootPathCost,
    ieee8021MstpRootPort,
    ieee8021MstpBridgePriority,
    ieee8021MstpVids0,
    ieee8021MstpVids1,
    ieee8021MstpVids2,
    ieee8021MstpVids3,
    ieee8021MstpRowStatus
}
STATUS      current
DESCRIPTION
    "Objects for the MST group"
::= { ieee8021MstpGroups 2 }

ieee8021MstpCistPortGroup OBJECT-GROUP
OBJECTS {
    ieee8021MstpCistPortUptime,
    ieee8021MstpCistPortAdminPathCost,
    ieee8021MstpCistPortDesignatedRoot,
    ieee8021MstpCistPortTopologyChangeAck,
    ieee8021MstpCistPortHelloTime,
    ieee8021MstpCistPortAdminEdgePort,
    ieee8021MstpCistPortOperEdgePort,
    ieee8021MstpCistPortMacEnabled,
    ieee8021MstpCistPortMacOperational,
    ieee8021MstpCistPortRestrictedRole,
    ieee8021MstpCistPortRestrictedTcn,
    ieee8021MstpCistPortRole,
    ieee8021MstpCistPortDisputed,
    ieee8021MstpCistPortCistRegionalRootId,
    ieee8021MstpCistPortCistPathCost,
    ieee8021MstpCistPortProtocolMigration,
    ieee8021MstpCistPortEnableBPDURx,
    ieee8021MstpCistPortEnableBPDUtx,
    ieee8021MstpCistPortPseudoRootId,
    ieee8021MstpCistPortIsL2Gp
}
STATUS      current
DESCRIPTION
    "Objects for the CIST Port group"
::= { ieee8021MstpGroups 3 }

ieee8021MstpPortGroup OBJECT-GROUP
OBJECTS {
    ieee8021MstpPortUptime,
    ieee8021MstpPortState,
    ieee8021MstpPortPriority,
    ieee8021MstpPortPathCost,
    ieee8021MstpPortDesignatedRoot,
    ieee8021MstpPortDesignatedCost,
    ieee8021MstpPortDesignatedBridge,
    ieee8021MstpPortDesignatedPort,
    ieee8021MstpPortRole,
```

```

        ieee8021MstpPortDisputed
    }
    STATUS      current
    DESCRIPTION
        "Objects for the MST Port group"
    ::= { ieee8021MstpGroups 4 }

ieee8021MstpFidToMstiGroup OBJECT-GROUP
    OBJECTS {
        ieee8021MstpFidToMstiMstId
    }
    STATUS      current
    DESCRIPTION
        "Objects for the MST FID to MSTID Allocation Table group"
    ::= { ieee8021MstpGroups 5 }

ieee8021MstpVlanGroup OBJECT-GROUP
    OBJECTS {
        ieee8021MstpVlanMstId
    }
    STATUS      current
    DESCRIPTION
        "Objects for the MST Configuration Table group"
    ::= { ieee8021MstpGroups 6 }

ieee8021MstpConfigIdGroup OBJECT-GROUP
    OBJECTS {
        ieee8021MstpConfigIdFormatSelector,
        ieee8021MstpConfigurationName,
        ieee8021MstpRevisionLevel,
        ieee8021MstpConfigurationDigest
    }
    STATUS      current
    DESCRIPTION
        "Objects for the MST Configuration Identifier group"
    ::= { ieee8021MstpGroups 7 }

ieee8021MstpCistPortExtensionGroup OBJECT-GROUP
    OBJECTS {
        ieee8021MstpCistPortAutoEdgePort,
        ieee8021MstpCistPortAutoIsolatePort
    }
    STATUS      current
    DESCRIPTION
        "Objects for the CIST Port Extension group
         for fragile bridges"
    ::= { ieee8021MstpGroups 8 }

-- =====
-- Compliance statements
-- =====

ieee8021MstpCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting Multiple
         Spanning Tree as defined in 13 of IEEE Std 802.1Q."

```

```
MODULE
  MANDATORY-GROUPS {
    ieee8021MstpCistGroup,
    ieee8021MstpGroup,
    ieee8021MstpCistPortGroup,
    ieee8021MstpPortGroup,
    ieee8021MstpFidToMstiGroup,
    ieee8021MstpVlanGroup,
    ieee8021MstpConfigIdGroup
  }

  GROUP ieee8021MstpCistPortExtensionGroup
  DESCRIPTION
    "Implementation of this group is optional."
  ::= { ieee8021MstpCompliances 1 }

END
```

17.7.7 Definitions for the IEEE8021-CFM MIB module

There are two modules for CFM as a result of the reindexing required with the addition of PBB (see Clause 17.3.7.2). Since the IEEE8021-CFM MIB module contains deprecated tables replaced by reindexed tables in IEEE8021-CFM-V2 MIB module, both modules are needed for CFM implementations.

17.7.7.1 Definitions for the IEEE8021-CFM MIB module

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

-- *****
-- IEEE 802.1ag (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       -- [IEEE Std 802.1AB-2005]
    VlanIdOrNone, VlanId   FROM Q-BRIDGE-MIB   -- [RFC4363]
    ieee802dot1mibs,
    IEEE8021VlanIndex      FROM IEEE8021-TC-MIB

;

ieee8021CfmMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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
              Postal: C/O IEEE 802.1 Working Group
                      IEEE Standards Association
                      445 Hoes Lane
                      P.O. Box 1331
                      Piscataway
                      NJ 08855-1331
                      USA
              E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG

    Contact: Norman Finn
              Postal: C/O IEEE 802.1 Working Group
```

IEEE Standards Association
445 Hoes Lane
P.O. Box 1331
Piscataway
NJ 08855-1331
USA

E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG

"

DESCRIPTION

"Connectivity Fault Management module.

Unless otherwise indicated, the references in this MIB module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.

This version of this MIB module is part of IEEE802.1Q; see the draft itself for full legal notices."

REVISION "201102270000Z" -- February 27, 2011

DESCRIPTION

"Addition of support for ICC format and minor edits as part of 2011 revision of IEEE Std 802.1Q."

REVISION "200811180000Z" -- November 18, 2008

DESCRIPTION

"Added new columns to the dot1agCfmMepTable to support new MEP functionality required for PBB-TE support. Modified dot1agCfmMepDbTable to support new functionality required for PBB-TE. Modified conformance clauses to indicate objects needed for PBB-TE support."

REVISION "200810150000Z" -- October 15, 2008

DESCRIPTION

"The IEEE8021-CFM-MIB Module was originally included in IEEE 802.1ag-2007. Some objects in this module are deprecated and replaced by objects in the IEEE8021-CFM-V2-MIB module defined in 802.1ap.

This revision is included in IEEE 802.1ap"

REVISION "200706100000Z" -- 06/10/2007 00:00GMT

DESCRIPTION

"Included in IEEE 802.1ag-2007

Copyright (C) IEEE802.1."

::= { ieee802dot1mibs 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 }

```

dotlagCfmConfigErrorList OBJECT IDENTIFIER ::= { dotlagMIBObjects 4 }
dotlagCfmMd          OBJECT IDENTIFIER ::= { dotlagMIBObjects 5 }
dotlagCfmMa          OBJECT IDENTIFIER ::= { dotlagMIBObjects 6 }
dotlagCfmMep          OBJECT IDENTIFIER ::= { dotlagMIBObjects 7 }

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

DotlagCfmMaintDomainNameType ::= TEXTUAL-CONVENTION
  STATUS      current
  DESCRIPTION
    "A value that represents a type (and thereby the format)
     of a DotlagCfmMaintDomainName. 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 DotlagCfmMaintDomainNameType
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 DotlagCfmMaintDomainNameType
objects and any dependent objects (e.g.,
DotlagCfmMaintDomainName objects) are consistent. An
inconsistentValue error MUST be generated if an attempt to
change an DotlagCfmMaintDomainNameType object would, for
example, lead to an undefined DotlagCfmMaintDomainName value.
In particular,
DotlagCfmMaintDomainNameType/DotlagCfmMaintDomainName pairs
MUST be changed together if the nameType changes.

"
REFERENCE
  "21.6.5, Table 21-19"
SYNTAX   INTEGER {
            none          (1),
            dnsLikeName   (2),
            macAddressAndUint (3),
            charString    (4)
}

```

{

DotlagCfmMaintDomainName ::= TEXTUAL-CONVENTION

STATUS current
DESCRIPTION

"Denotes a generic Maintenance Domain Name.

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

The value of a DotlagCfmMaintDomainName object MUST always be consistent with the value of the associated DotlagCfmMaintDomainNameType object. Attempts to set an DotlagCfmMaintDomainName object to a value inconsistent with the associated DotlagCfmMaintDomainNameType 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 DotlagCfmMaintDomainNameType object means that no Maintenance Domain name is present, and the contents of the DotlagCfmMaintDomainName object are meaningless.

See the DESCRIPTION of the DotlagCfmMaintAssocNameType TEXTUAL-CONVENTION for a discussion of the length limits on the Maintenance Domain name and Maintenance Association name.

"

REFERENCE

"21.6.5"

SYNTAX OCTET STRING (SIZE(1..43))

DotlagCfmMaintAssocNameType ::= TEXTUAL-CONVENTION

STATUS current
DESCRIPTION

"A value that represents a type (and thereby the format) of a DotlagCfmMaintAssocName. 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		(LSB)				
+	-	+	-	+	-	+	-	+

ieeeReserved(xx) Reserved for definition by IEEE 802.1
 xx values can be [5..31] and [64..255]
 ICCformat(32) ICC-based format as specified in ITU-T Y.1731
 ituReserved(xx) Reserved for definition by ITU-T Y.1731
 xx values range from [33..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 DotlagCfmMaintDomainNameType object contains none(1), then the DotlagCfmMaintAssocName object MUST be 45 octets or less in length. Otherwise, the length of the DotlagCfmMaintDomainName object plus the length of the DotlagCfmMaintAssocName object, added together, MUST total less than or equal to 44 octets.

"

REFERENCE

"21.6.5.4, Table 21-20"

SYNTAX INTEGER {
 primaryVid (1),
 charString (2),
 unsignedInt16 (3),
 rfc2865VpnId (4),
 ICCformat (32)
}

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

The value of a DotlagCfmMaintAssocName object MUST always be consistent with the value of the associated DotlagCfmMaintAssocNameType object. Attempts to set an DotlagCfmMaintAssocName object to a value inconsistent with the associated DotlagCfmMaintAssocNameType 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 the 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

"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

"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

"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

"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

"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.
defMFHdefault(2)   MHFs can be created on this VID on any
                  Bridge port through which this VID can
                  pass.
defMFHexplicit(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.
defMFHdefer(4)     The creation of MHFs is determined by the
                  corresponding Maintenance Domain variable
                  (dot1agCfmMaCompMhfCreation).
"
```

REFERENCE**"12.14.5.1.3:c and 22.2.3"**

```

SYNTAX      INTEGER {
            defMHFnone      (1),
            defMFHdefault   (2),
            defMFHexplicit  (3),
            defMFHdefer     (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**"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 different 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

"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:

bDefRDI(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

"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
 "12.3 1)"
 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 deprecated
 DESCRIPTION
 "There is one CFM Stack table per bridge. It permits the retrieval of information about the Maintenance Points configured on any given interface.
 **NOTE: this object is deprecated due to re-indexing of the table."
 "
 REFERENCE
 "802.1ag clauses 12.14.2"
 ::= { dot1agCfmStack 1 }

dot1agCfmStackEntry OBJECT-TYPE
 SYNTAX Dot1agCfmStackEntry
 MAX-ACCESS not-accessible
 STATUS deprecated
 DESCRIPTION
 "The Stack table entry
 **NOTE: this object is deprecated due to re-indexing of the table."
 "
 INDEX { dot1agCfmStackIfIndex, dot1agCfmStackVlanIdOrNone,
 dot1agCfmStackMdLevel, dot1agCfmStackDirection }

```

        }
 ::= { dotlagCfmStackTable 1 }

DotlagCfmStackEntry ::= SEQUENCE {
    dotlagCfmStackIfIndex      InterfaceIndex,
    dotlagCfmStackVlanIdOrNone VlanIdOrNone,
    dotlagCfmStackMdLevel     DotlagCfmMDLevel,
    dotlagCfmStackDirection   DotlagCfmMpDirection,
    dotlagCfmStackMdIndex     Unsigned32,
    dotlagCfmStackMaIndex     Unsigned32,
    dotlagCfmStackMepId       DotlagCfmMepIdOrZero,
    dotlagCfmStackMacAddress  MacAddress
}

dotlagCfmStackIfIndex OBJECT-TYPE
SYNTAX      InterfaceIndex
MAX-ACCESS  not-accessible
STATUS      deprecated
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
dotlagCfmStackTable, 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
dotlagCfmStackTable with the interface index.

**NOTE: this object is deprecated due to re-indexing of
the table.

**NOTE: this object is deprecated due to re-indexing of the
table.

"
REFERENCE
"12.14.2.1.2:a"
 ::= { dotlagCfmStackEntry 1 }

dotlagCfmStackVlanIdOrNone OBJECT-TYPE
SYNTAX      VlanIdOrNone
MAX-ACCESS  not-accessible
STATUS      deprecated
DESCRIPTION
"VLAN ID to which the MP is attached, or 0, if none.

**NOTE: this object is deprecated due to re-indexing of the
table.

"
REFERENCE
"802.1ag clauses 12.14.2.1.2:d, 22.1.7"
 ::= { dotlagCfmStackEntry 2 }

dotlagCfmStackMdLevel OBJECT-TYPE
SYNTAX      DotlagCfmMDLevel
MAX-ACCESS  not-accessible
STATUS      deprecated
DESCRIPTION
"MD Level of the Maintenance Point.

**NOTE: this object is deprecated due to re-indexing of the
table.

```

```
"  
REFERENCE  
    "12.14.2.1.2:b"  
::= { dotlagCfmStackEntry 3 }  
  
dotlagCfmStackDirection OBJECT-TYPE  
    SYNTAX      DotlagCfmMpDirection  
    MAX-ACCESS  not-accessible  
    STATUS     deprecated  
    DESCRIPTION  
        "Direction in which the MP faces on the Bridge Port  
        **NOTE: this object is deprecated due to re-indexing of the  
        table."  
        "  
REFERENCE  
    "12.14.2.1.2:c"  
::= { dotlagCfmStackEntry 4 }  
  
dotlagCfmStackMdIndex OBJECT-TYPE  
    SYNTAX      Unsigned32  
    MAX-ACCESS  read-only  
    STATUS     deprecated  
    DESCRIPTION  
        "The index of the Maintenance Domain in the dotlagCfmMdTable  
        to which the MP is associated, or 0, if none."  
        "  
REFERENCE  
    "12.14.2.1.3:b"  
::= { dotlagCfmStackEntry 5 }  
  
dotlagCfmStackMaIndex OBJECT-TYPE  
    SYNTAX      Unsigned32  
    MAX-ACCESS  read-only  
    STATUS     deprecated  
    DESCRIPTION  
        "The index of the MA in the dotlagCfmMaNetTable and  
        dotlagCfmMaCompTable to which the MP is associated, or 0, if  
        none.  
        **NOTE: this object is deprecated due to re-indexing of the  
        table."  
        "  
REFERENCE  
    "12.14.2.1.3:c"  
::= { dotlagCfmStackEntry 6 }  
  
dotlagCfmStackMepId OBJECT-TYPE  
    SYNTAX      DotlagCfmMepIdOrZero  
    MAX-ACCESS  read-only  
    STATUS     deprecated  
    DESCRIPTION  
        "If an MEP is configured, the MEPID, else 0"  
REFERENCE  
    "12.14.2.1.3:d  
    **NOTE: this object is deprecated due to re-indexing of the  
    table."  
    "  
::= { dotlagCfmStackEntry 7 }  
  
dotlagCfmStackMacAddress OBJECT-TYPE
```

```

SYNTAX      MacAddress
MAX-ACCESS  read-only
STATUS      deprecated
DESCRIPTION
  "MAC address of the MP.
   **NOTE: this object is deprecated due to re-indexing of the
table.
"
REFERENCE
  "12.14.2.1.3:e"
 ::= { dotlagCfmStackEntry 8 }

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

dotlagCfmVlanTable OBJECT-TYPE
SYNTAX      SEQUENCE OF DotlagCfmVlanEntry
MAX-ACCESS  not-accessible
STATUS      deprecated
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.
**NOTE: this object is deprecated due to re-indexing of the
table.
"
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."
 ::= { dotlagCfmVlan 1 }

dotlagCfmVlanEntry OBJECT-TYPE
SYNTAX      DotlagCfmVlanEntry
MAX-ACCESS  not-accessible
STATUS      deprecated
DESCRIPTION
  "The VLAN table entry.
   **NOTE: this object is deprecated due to re-indexing of the
table.
"
INDEX { dotlagCfmVlanComponentId, dotlagCfmVlanVid }
 ::= { dotlagCfmVlanTable 1 }

DotlagCfmVlanEntry ::= SEQUENCE {
  dotlagCfmVlanComponentId DotlagCfmPbbComponentIdentifier,

```

```
dotlagCfmVlanVid          VlanId,
dotlagCfmVlanPrimaryVid   VlanId,
dotlagCfmVlanRowStatus    RowStatus
}

dotlagCfmVlanComponentId OBJECT-TYPE
SYNTAX      DotlagCfmPbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      deprecated
DESCRIPTION
"The bridge component within the system to which the information
in this dotlagCfmVlanEntry 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.
**NOTE: this object is deprecated due to re-indexing of the
table.
"
REFERENCE
"12.3 1)"
::= { dotlagCfmVlanEntry 1 }

dotlagCfmVlanVid OBJECT-TYPE
SYNTAX      VlanId
MAX-ACCESS  not-accessible
STATUS      deprecated
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.
**NOTE: this object is deprecated due to re-indexing of the
table.
"
::= { dotlagCfmVlanEntry 2 }

dotlagCfmVlanPrimaryVid OBJECT-TYPE
SYNTAX      VlanId
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
"This is the Primary VLAN ID of the VLAN with which this
entry's dotlagCfmVlanVid is associated. This value MUST not
equal the value of dotlagCfmVlanVid.
**NOTE: this object is deprecated due to re-indexing of the
table.
"
::= { dotlagCfmVlanEntry 3 }

dotlagCfmVlanRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      deprecated
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.
**NOTE: this object is deprecated due to re-indexing of the
table.
```

```

"
 ::= { dotlagCfmVlanEntry 4 }

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

dotlagCfmDefaultMdDefLevel OBJECT-TYPE
    SYNTAX      DotlagCfmMDLevel
    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 dotlagCfmDefaultMdEntry whose
         dotlagCfmDefaultMdLevel object contains the value -1.

        After this initialization, this object needs to be persistent
        upon reboot or restart of a device.
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
    REFERENCE
        "12.14.3.1.3:c, 12.14.3.2.2:b"
    DEFVAL {0}
    ::= { dotlagCfmDefaultMd 1 }

dotlagCfmDefaultMdDefMhfCreation OBJECT-TYPE
    SYNTAX      DotlagCfmMhfCreation {
        defMHFnone      (1),
        defMFHdefault   (2),
        defMFHexplicit  (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
         dotlagCfmDefaultMdEntry whose dotlagCfmDefaultMdMhfCreation
         object contains the value defMFHdefer. Since, in this
         variable, there is no encompassing Maintenance Domain, the
         value defMFHdefer is not allowed.

        After this initialization, this object needs to be persistent
        upon reboot or restart of a device.
        "
    REFERENCE
        "12.14.3.1.3:d"
    DEFVAL {defMHFnone}
    ::= { dotlagCfmDefaultMd 2 }

dotlagCfmDefaultMdDefIdPermission OBJECT-TYPE
    SYNTAX      DotlagCfmIdPermission {
        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
  dotlagCfmDefaultMdEntry whose dotlagCfmDefaultMdIdPermission
  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
  "12.14.3.1.3:e"
DEFVAL { sendIdNone }
::= { dotlagCfmDefaultMd 3 }

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

dotlagCfmDefaultMdTable OBJECT-TYPE
SYNTAX      SEQUENCE OF DotlagCfmDefaultMdEntry
MAX-ACCESS  not-accessible
STATUS      deprecated
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 dotlagCfmDefaultMdStatus 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.

  **NOTE: this object is deprecated due to re-indexing of the
  table.
"
REFERENCE
  " 12.14.3"
::= { dotlagCfmDefaultMd 4 }
```

```

dot1agCfmDefaultMdEntry OBJECT-TYPE
    SYNTAX      Dot1agCfmDefaultMdEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The Default MD Level table entry.
         **NOTE: this object is deprecated due to re-indexing of the
         table.
         "
        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      deprecated
    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.
         **NOTE: this object is deprecated due to re-indexing of the
         table.
         "
        REFERENCE
            "12.3 1)"
        ::= { dot1agCfmDefaultMdEntry 1 }

dot1agCfmDefaultMdPrimaryVid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The Primary VID of the VLAN to which this entry's objects
         apply.
         **NOTE: this object is deprecated due to re-indexing of the
         table.
         "
        ::= { dot1agCfmDefaultMdEntry 2 }

dot1agCfmDefaultMdStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      deprecated
    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.

```

```
    **NOTE: this object is deprecated due to re-indexing of the
    table.
    "
REFERENCE
    "12.14.3.1.3:b"
::= { dotlagCfmDefaultMdEntry 3 }

dotlagCfmDefaultMdLevel OBJECT-TYPE
    SYNTAX      DotlagCfmMDLevelOrNone
    MAX-ACCESS  read-write
    STATUS      deprecated
    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
         dotlagCfmDefaultMdDefLevel.

    **NOTE: this object is deprecated due to re-indexing of the
    table.
    "
REFERENCE
    "12.14.3.1.3:c, 12.14.3.2.2:b"
DEFVAL {-1}
::= { dotlagCfmDefaultMdEntry 4 }

dotlagCfmDefaultMdMhfCreation OBJECT-TYPE
    SYNTAX      DotlagCfmMhfCreation
    MAX-ACCESS  read-write
    STATUS      deprecated
    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 dotlagCfmDefaultMdDefMhfCreation.

The value of this variable is meaningless if the values of
dotlagCfmDefaultMdStatus is false.

    **NOTE: this object is deprecated due to re-indexing of the
    table.
    "
REFERENCE
    "12.14.3.1.3:d"
DEFVAL {defMhfdefer}
::= { dotlagCfmDefaultMdEntry 5 }

dotlagCfmDefaultMdIdPermission OBJECT-TYPE
    SYNTAX      DotlagCfmIdPermission
    MAX-ACCESS  read-write
    STATUS      deprecated
    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 dotlagCfmDefaultMdDefIdPermission.

The value of this variable is meaningless if the values of
dotlagCfmDefaultMdStatus is false.
```

```

    **NOTE: this object is deprecated due to re-indexing of the
    table.
    "
REFERENCE
    "12.14.3.1.3:e"
DEFVAL { sendIdDefer }
 ::= { dotlagCfmDefaultMdEntry 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
-- ****

dotlagCfmConfigErrorListTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF DotlagCfmConfigErrorListEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The CFM Configuration Error List table provides a list of
         Interfaces and VIDs that are incorrectly configured.
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
REFERENCE
    "12.14.4"
 ::= {dotlagCfmConfigErrorList 1}

dotlagCfmConfigErrorListEntry OBJECT-TYPE
    SYNTAX      DotlagCfmConfigErrorListEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The Config Error List Table entry
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
INDEX { dotlagCfmConfigErrorListVid,
         dotlagCfmConfigErrorListIfIndex
     }
 ::= { dotlagCfmConfigErrorListTable 1}

DotlagCfmConfigErrorListEntry ::= SEQUENCE {
    dotlagCfmConfigErrorListVid          VlanId,
    dotlagCfmConfigErrorListIfIndex      InterfaceIndex,
    dotlagCfmConfigErrorListErrorType   DotlagCfmConfigErrors
}

dotlagCfmConfigErrorListVid OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The VLAN ID of the VLAN with interfaces in error.

```

```
    **NOTE: this object is deprecated due to re-indexing of the
    table.
    "
REFERENCE
    "12.14.4.1.2:a"
::= { dotlagCfmConfigErrorListEntry 1 }

dotlagCfmConfigErrorListIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    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
dotlagCfmConfigErrorListTable indexed by that
InterfaceIndex value.

    **NOTE: this object is deprecated due to re-indexing of the
table.
    "
REFERENCE
    "12.14.4.1.2:b"
::= { dotlagCfmConfigErrorListEntry 2 }

dotlagCfmConfigErrorListErrorType OBJECT-TYPE
    SYNTAX      DotlagCfmConfigErrors
    MAX-ACCESS  read-only
    STATUS      deprecated
    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.

    **NOTE: this object is deprecated due to re-indexing of the
table.
    "
REFERENCE
    "12.14.4.1.3:b"
::= { dotlagCfmConfigErrorListEntry 3 }

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

dotlagCfmMdTableNextIndex OBJECT-TYPE
    SYNTAX      Dot1afCfmIndexIntegerNextFree
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains an unused value for dotlagCfmMdIndex in
        the dotlagCfmMdTable, or a zero to indicate that none exist.
```

```

"
 ::= { dotlagCfmMd 1 }

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

dotlagCfmMdTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF DotlagCfmMdEntry
    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"
 ::= { dotlagCfmMd 2 }

dotlagCfmMdEntry OBJECT-TYPE
    SYNTAX      DotlagCfmMdEntry
    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 {dotlagCfmMdIndex }
 ::= { dotlagCfmMdTable 1 }

DotlagCfmMdEntry ::= SEQUENCE {
    dotlagCfmMdIndex          Unsigned32,
    dotlagCfmMdFormat          DotlagCfmMaintDomainNameType,
    dotlagCfmMdName             DotlagCfmMaintDomainName,
    dotlagCfmMdMdLevel          DotlagCfmMDLevel,
    dotlagCfmMdMhfCreation       DotlagCfmMhfCreation,
    dotlagCfmMdMhfIdPermission   DotlagCfmIdPermission,
    dotlagCfmMdMaNextIndex        DotlafCfmIndexIntegerNextFree,
    dotlagCfmMdRowStatus          RowStatus
}

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

        dotlagCfmMdTableNextIndex 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.

"

`::= { dotlagCfmMdEntry 1 }`

dotlagCfmMdFormat OBJECT-TYPE
SYNTAX DotlagCfmMaintDomainNameType
MAX-ACCESS read-create
STATUS current
DESCRIPTION "The type (and thereby format) of the Maintenance Domain Name."
REFERENCE "21.6.5.1"
DEFVAL { charString }
`::= { dotlagCfmMdEntry 2 }`

dotlagCfmMdName OBJECT-TYPE
SYNTAX DotlagCfmMaintDomainName
MAX-ACCESS read-create
STATUS current
DESCRIPTION "The Maintenance Domain name. The type/format of this object is determined by the value of the dotlagCfmMdNameType 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" }
`::= { dotlagCfmMdEntry 3 }`

dotlagCfmMdMdLevel OBJECT-TYPE
SYNTAX DotlagCfmMDLevel
MAX-ACCESS read-create
STATUS current
DESCRIPTION "The Maintenance Domain Level."
REFERENCE "12.14.5.1.3:b"
DEFVAL { 0 }
`::= { dotlagCfmMdEntry 4 }`

dotlagCfmMdMhfCreation OBJECT-TYPE
SYNTAX DotlagCfmMhfCreation {
 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
  "12.14.5.1.3:c"
DEFVAL { defMHFnone }
 ::= { dotlagCfmMdEntry 5 }

dotlagCfmMdMhfIdPermission OBJECT-TYPE
SYNTAX      DotlagCfmIdPermission {
    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
  "12.14.5.1.3:d"
DEFVAL { sendIdNone }
 ::= { dotlagCfmMdEntry 6 }

dotlagCfmMdMaNextIndex 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 dotlagCfmMaNetTable and the dotlagCfmMaCompTable, for
  this Maintenance Domain when the management entity wants to
  create a new row in those tables.
  "
 ::= { dotlagCfmMdEntry 7 }

dotlagCfmMdRowStatus 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.
  "
 ::= { dotlagCfmMdEntry 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
-- ****
```

```
dotlagCfmMaNetTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF DotlagCfmMaNetEntry
    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 dotlagCfmMaCompTable.

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 3.19 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 dotlagCfmMaCompEntry for the same MA.

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

```
""
REFERENCE
    "18.2"
::= { dotlagCfmMa 1 }
```

```
dotlagCfmMaNetEntry OBJECT-TYPE
    SYNTAX      DotlagCfmMaNetEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The MA table entry."
INDEX {dotlagCfmMdIndex, dotlagCfmMaIndex }
::= { dotlagCfmMaNetTable 1 }
```

```

DotlagCfmMaNetEntry ::= SEQUENCE {
    dotlagCfmMaIndex                  Unsigned32,
    dotlagCfmMaNetFormat              DotlagCfmMaintAssocNameType,
    dotlagCfmMaNetName                DotlagCfmMaintAssocName,
    dotlagCfmMaNetCcmInterval         DotlagCfmCcmInterval,
    dotlagCfmMaNetRowStatus           RowStatus
}

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

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

dotlagCfmMaNetFormat OBJECT-TYPE
SYNTAX      DotlagCfmMaintAssocNameType
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"
::= { dotlagCfmMaNetEntry 2 }

dotlagCfmMaNetName OBJECT-TYPE
SYNTAX      DotlagCfmMaintAssocName
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
     dotlagCfmMaNetNameType object. This name MUST be unique within
     a maintenance domain.
    "
REFERENCE

    "802.1ag clauses 21.6.5.6, and Table 21-20"
::= { dotlagCfmMaNetEntry 3 }

dotlagCfmMaNetCcmInterval OBJECT-TYPE
SYNTAX      DotlagCfmCcmInterval
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION

    "Interval between CCM transmissions to be used by all MEPs
     in the MA.
    "
REFERENCE

    "12.14.6.1.3:e"
DEFVAL { intervals }
::= { dotlagCfmMaNetEntry 4 }

dotlagCfmMaNetRowStatus 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.

    "
::= { dotlagCfmMaNetEntry 5 }

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

dotlagCfmMaCompTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF DotlagCfmMaCompEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    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 dotlagCfmMaNetTable.

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

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

        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
REFERENCE
    "18.2"
::= { dotlagCfmMa 2 }

dotlagCfmMaCompEntry OBJECT-TYPE
    SYNTAX      DotlagCfmMaCompEntry
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The MA table entry.
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
INDEX {dotlagCfmMaComponentId,
       dotlagCfmMdIndex, dotlagCfmMaIndex }
```

```

 ::= { dotlagCfmMaCompTable 1 }

DotlagCfmMaCompEntry ::= SEQUENCE {
    dotlagCfmMaComponentId          DotlagCfmPbbComponentIdentifier,
    dotlagCfmMaCompPrimaryVlanId    VlanIdOrNone,
    dotlagCfmMaCompMhfCreation      DotlagCfmMhfCreation,
    dotlagCfmMaCompIdPermission     DotlagCfmIdPermission,
    dotlagCfmMaCompNumberOfVids    Unsigned32,
    dotlagCfmMaCompRowStatus        RowStatus
}

dotlagCfmMaComponentId OBJECT-TYPE
    SYNTAX      DotlagCfmPbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      deprecated
    DESCRIPTION
        "The bridge component within the system to which the information
         in this dotlagCfmMaCompEntry 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.
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
    REFERENCE
        "12.3 1)"
    ::= { dotlagCfmMaCompEntry 1 }

dotlagCfmMaCompPrimaryVlanId OBJECT-TYPE
    SYNTAX      VlanIdOrNone
    MAX-ACCESS  read-create
    STATUS      deprecated
    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
         dotlagCfmVlanTable lists them.
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
    REFERENCE
        "12.14.6.1.3:b"
    ::= { dotlagCfmMaCompEntry 2 }

dotlagCfmMaCompMhfCreation OBJECT-TYPE
    SYNTAX      DotlagCfmMhfCreation
    MAX-ACCESS  read-create
    STATUS      deprecated
    DESCRIPTION
        "Indicates if the Management entity can create MHFs (MIP Half
         Function) for this MA.
        **NOTE: this object is deprecated due to re-indexing of the
        table.
        "
    REFERENCE
        "12.14.6.1.3:c"
    DEFVAL { defMHFdefer }
    ::= { dotlagCfmMaCompEntry 3 }

dotlagCfmMaCompIdPermission OBJECT-TYPE

```

```
SYNTAX      Dot1agCfmIdPermission
MAX-ACCESS  read-create
STATUS      deprecated
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.
    **NOTE: this object is deprecated due to re-indexing of the
table.
"
REFERENCE
    "12.14.6.1.3:d"
DEFVAL { sendIdDefer }
::= { dot1agCfmMaCompEntry 4 }

dot1agCfmMaCompNumberOfVids OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-create
STATUS      deprecated
DESCRIPTION
    "The number of VIDs associated with the MA.
    **NOTE: this object is deprecated due to re-indexing of the
table.
"
REFERENCE
    "12.14.6.1.3:b"
::= { dot1agCfmMaCompEntry 5 }

dot1agCfmMaCompRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      deprecated
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.
    **NOTE: this object is deprecated due to re-indexing of the
table.
"
::= { 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
"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
"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 3.19).

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 false. 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 false.
 - 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 dotlagCfmMepTransmitLbmStatus value from false to true 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 dotlagCfmMepTransmitLbmStatus 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. Setting this MIB object does not set the corresponding LBIactive state machine variable.
- 3) Setup the different data to be sent (number of messages, optional TLVs,...), except do not set dotlagCfmMepTransmitLbmMessages.
- 4) Record the current values of dotlagCfmMepLbrIn, dotlagCfmMepLbrInOutOfOrder, and dotlagCfmMepLbrBadMsdu.
- 6) Set dotlagCfmMepTransmitLbmMessages to a non-zero value to initiate transmission of Loopback messages. The dotlagCfmMepTransmitLbmMessages indicates the number of LBMs to be sent and is not decremented as loopbacks are actually sent. dotlagCfmMepTransmitLbmMessages is not equivalent to the LBMsToSend state machine variable.
- 7) Check the value of dotlagCfmMepTransmitLbmResultOK to find out if the operation was successfully initiated or not.
- 8) Monitor the value of dotlagCfmMepTransmitLbmStatus. When it is reset to false, the last LBM has been transmitted. Wait an additional 5 seconds to ensure that all LBRs have been returned.
- 9) Compare dotlagCfmMepLbrIn, dotlagCfmMepLbrInOutOfOrder, and dotlagCfmMepLbrBadMsdu to their old values from step 4, above, to get the results of the test.

*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 dotlagCfmMepTransmitLtmSeqNumber and dotlagCfmMepTransmitLtmEgressIdentifier return the information required to recover the results of the LTM from the dotlagCfmLtrTable.

"

REFERENCE

"802.1ag clauses 12.14.7 and 19.2"

::= { dotlagCfmMep 1 }

```
dotlagCfmMepEntry OBJECT-TYPE
  SYNTAX      DotlagCfmMepEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The MEP table entry"
  INDEX { dotlagCfmMdIndex,
          dotlagCfmMaIndex,
          dotlagCfmMepIdentifier
        }
  ::= { dotlagCfmMepTable 1 }
```

```

Dot1agCfmMepEntry ::= SEQUENCE {
    dot1agCfmMepIdentifier
    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
    dot1agCfmMepPbbTeCanReportPbbTePresence
    dot1agCfmMepPbbTeTrafficMismatchDefect
    dot1agCfmMepPbbTransmitLbmLtmReverseVid
    dot1agCfmMepPbbTeMismatchAlarm
    dot1agCfmMepPbbTeLocalMismatchDefect
    dot1agCfmMepPbbTeMismatchSinceReset
}

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"

::= { dotlagCfmMepEntry 1 }

dotlagCfmMepIfIndex 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

"12.14.7.1.3:b"

::= { dotlagCfmMepEntry 2 }

dotlagCfmMepDirection OBJECT-TYPE

SYNTAX DotlagCfmMpDirection

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"

::= { dotlagCfmMepEntry 3 }

dotlagCfmMepPrimaryVid 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 }

::= { dotlagCfmMepEntry 4 }

dotlagCfmMepActive 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 }
 ::= { dotlagCfmMepEntry 5 }

dotlagCfmMepFngState OBJECT-TYPE
SYNTAX DotlagCfmFngState
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 }
 ::= { dotlagCfmMepEntry 6 }

dotlagCfmMepCciEnabled 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 }
 ::= { dotlagCfmMepEntry 7 }

dotlagCfmMepCcmLtmPriority 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
"12.14.7.1.3:h"
 ::= { dotlagCfmMepEntry 8 }

dotlagCfmMepMacAddress OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"MAC address of the MEP."

```
REFERENCE
    "12.14.7.1.3:i and 19.4"
 ::= { dotlagCfmMepEntry 9 }

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

dotlagCfmMepFngAlarmTime 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 }
 ::= { dotlagCfmMepEntry 11 }

dotlagCfmMepFngResetTime 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 }
 ::= { dotlagCfmMepEntry 12 }

dotlagCfmMepHighestPrDefect OBJECT-TYPE
    SYNTAX      DotlagCfmHighestDefectPri
    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
        "12.14.7.1.3:n 20.33.9 and Table 21-1"
 ::= { dotlagCfmMepEntry 13 }

dotlagCfmMepDefects OBJECT-TYPE
    SYNTAX      DotlagCfmMepDefects
    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."
 ::= { dotlagCfmMepEntry 14 }

dotlagCfmMepErrorCcmLastFailure 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"
 ::= { dotlagCfmMepEntry 15 }

dotlagCfmMepXconCcmLastFailure 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"
 ::= { dotlagCfmMepEntry 16 }

dotlagCfmMepCcmSequenceErrors 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"
 ::= { dotlagCfmMepEntry 17 }

dotlagCfmMepCciSentCcms 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"
 ::= { dotlagCfmMepEntry 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
        "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
        "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
        "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
        "12.14.7.1.3:ab and 20.36.1"
```

```
 ::= { dotlagCfmMepEntry 23 }

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

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

dotlagCfmMepTransmitLbmStatus OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "A Boolean flag set to true by the MEP Loopback Initiator State
        Machine or an MIB manager to indicate
        that another LBM is being transmitted.
        Reset to false by the MEP Loopback Initiator State Machine."
    DEFVAL { false }
 ::= { dotlagCfmMepEntry 26 }

dotlagCfmMepTransmitLbmDestMacAddress 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
        dotlagCfmMepTransmitLbmDestIsMepId is 'false'.
        "
    REFERENCE
        "12.14.7.3.2:b"
 ::= { dotlagCfmMepEntry 27 }

dotlagCfmMepTransmitLbmDestMepId OBJECT-TYPE
    SYNTAX      DotlagCfmMepIdOrZero
    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
        dotlagCfmMepTransmitLbmDestIsMepId is 'true'.
```

```

"
REFERENCE
"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
"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
"12.14.7.3.2:c"
DEFVAL { 1 }
 ::= {dot1agCfmMepEntry 30 }

dot1agCfmMepTransmitLbmDataTlv OBJECT-TYPE
SYNTAX      OCTET STRING
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. The intent is to be able
to fill the frame carrying the CFM PDU to its maximum length.
This may lead to fragmentation in some cases.
"
REFERENCE
"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
"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
  "12.14.7.3.2:e"
DEFVAL { false }
 ::= { dotlagCfmMepEntry 33 }

dotlagCfmMepTransmitLbmResultOK 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
  "12.14.7.3.3:a"
DEFVAL { true }
 ::= { dotlagCfmMepEntry 34 }

dotlagCfmMepTransmitLbmSeqNumber OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "The Loopback Transaction Identifier
  (dotlagCfmMepNextLbmTransId) of the first LBM (to be) sent.
  The value returned is undefined if
  dotlagCfmMepTransmitLbmResultOK is false.
  "
REFERENCE
  "12.14.7.3.3:a"
 ::= { dotlagCfmMepEntry 35 }

dotlagCfmMepTransmitLtmStatus 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 LTM may be transmitted.
  Reset to false by the MEP Linktrace Initiator State Machine."
DEFVAL { true }
 ::= { dotlagCfmMepEntry 36 }

dotlagCfmMepTransmitLtmFlags OBJECT-TYPE
SYNTAX      BITS {
```

```

        useFDOnly      (0)
    }
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The flags field for LTMs transmitted by the MEP."
REFERENCE
    "12.14.7.4.2:b and 20.37.1"
DEFVAL { {useFDOnly} } 
 ::= { dotlagCfmMepEntry 37 }

dotlagCfmMepTransmitLtmTargetMacAddress 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
dotlagCfmMepTransmitLtmTargetIsMepId is 'false'.
"
REFERENCE
    "12.14.7.4.2:c"
 ::= { dotlagCfmMepEntry 38 }

dotlagCfmMepTransmitLtmTargetMepId OBJECT-TYPE
SYNTAX      DotlagCfmMepIdOrZero
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
dotlagCfmMepTransmitLtmTargetIsMepId is 'true'.
"
REFERENCE
    "12.14.7.4.2:c"
 ::= { dotlagCfmMepEntry 39 }

dotlagCfmMepTransmitLtmTargetIsMepId 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
    "12.14.7.4.2:c"
 ::= { dotlagCfmMepEntry 40 }

dotlagCfmMepTransmitLtmTtl 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. Decrement 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

"12.14.7.4.2:d and 21.8.4"

DEFVAL { 64 }

::= { dotlagCfmMepEntry 41 }

dotlagCfmMepTransmitLtmResult 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

"12.14.7.4.3:a"

DEFVAL { true }

::= { dotlagCfmMepEntry 42 }

dotlagCfmMepTransmitLtmSeqNumber OBJECT-TYPE

SYNTAX Unsigned32

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The LTM Transaction Identifier (dotlagCfmMepLtmNextSeqNumber) of the LTM sent. The value returned is undefined if dotlagCfmMepTransmitLtmResult is false."

"

REFERENCE

"12.14.7.4.3:a"

::= { dotlagCfmMepEntry 43 }

dotlagCfmMepTransmitLtmEgressIdentifier 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 dotlagCfmMepTransmitLtmResult is false.

"

REFERENCE

"12.14.7.4.3:b and 21.8.8"
 ::= { dotlagCfmMepEntry 44 }

dotlagCfmMepRowStatus 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.

"

::= { dotlagCfmMepEntry 45 }

dotlagCfmMepPbbTeCanReportPbbTePresence OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"A Boolean valued parameter that is set to true if the system has the capability to report the presence of traffic and that the capability is enabled. Traffic presence reporting is an optional PBB-TE feature."

REFERENCE

"12.14.7.1.3:af and 21.6.1.4"

DEFVAL { false }

::= { dotlagCfmMepEntry 46 }

dotlagCfmMepPbbTeTrafficMismatchDefect OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"A Boolean valued parameter that is set to true if the system has detected a traffic field mismatch defect. Mismatch detection is an optional PBB-TE feature."

REFERENCE

"12.14.7.1.3:ah and 21.6.1.4"

::= { dotlagCfmMepEntry 47 }

dotlagCfmMepPbbTransmitLbmLtmReverseVid OBJECT-TYPE

SYNTAX IEEE8021VlanIndex

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This column specifies the value to use in the Reverse VID value field of PBB-TE MIP TLVs contained within TransmitLTM pdus."

REFERENCE

"12.14.7.4.2"

::= { dotlagCfmMepEntry 48 }

```
dotlagCfmMepPbbTeMismatchAlarm OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS     read-create
    STATUS         current
    DESCRIPTION
        "A Boolean valued parameter that is set to true if the system
         is to allow a mismatch defect to generate a fault alarm."
    REFERENCE
        "12.14.7.1.3:ag and 21.6.1.4"
    DEFVAL { false }
    ::= { dotlagCfmMepEntry 49 }
```

```
dotlagCfmMepPbbTeLocalMismatchDefect OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS     read-only
    STATUS         current
    DESCRIPTION
        "A Boolean valued parameter that is set to true if the system
         has detected a local mismatch defect. Mismatch detection
         is an optional PBB-TE feature."
    REFERENCE
        "12.14.7.1.3:ai and 21.6.1.4"
    ::= { dotlagCfmMepEntry 50 }
```

```
dotlagCfmMepPbbTeMismatchSinceReset OBJECT-TYPE
    SYNTAX          TruthValue
    MAX-ACCESS     read-only
    STATUS         current
    DESCRIPTION
        "A Boolean valued parameter indicating if the mismatch defect
         has been present since the MEP Mismatch Fault Notification
         Generator was last in the MFNG_RESET state."
    REFERENCE
        "12.14.7.1.3:aj"
    ::= { dotlagCfmMepEntry 51 }
```

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

```
dotlagCfmLtrTable OBJECT-TYPE
    SYNTAX          SEQUENCE OF DotlagCfmLtrEntry
    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
        "12.14.7.5"
    ::= { dotlagCfmMep 2 }
```

```

dotlagCfmLtrEntry OBJECT-TYPE
  SYNTAX      DotlagCfmLtrEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The Linktrace Reply table entry."
  INDEX { dotlagCfmMdIndex,
           dotlagCfmMaIndex,
           dotlagCfmMepIdentifier,
           dotlagCfmLtrSeqNumber,
           dotlagCfmLtrReceiveOrder
         }
 ::= { dotlagCfmLtrTable 1 }

DotlagCfmLtrEntry ::= SEQUENCE {
  dotlagCfmLtrSeqNumber          Unsigned32,
  dotlagCfmLtrReceiveOrder       Unsigned32,
  dotlagCfmLtrTtl               Unsigned32,
  dotlagCfmLtrForwarded         TruthValue,
  dotlagCfmLtrTerminalMep       TruthValue,
  dotlagCfmLtrLastEgressIdentifier OCTET STRING,
  dotlagCfmLtrNextEgressIdentifier OCTET STRING,
  dotlagCfmLtrRelay              DotlagCfmRelayActionFieldValue,
  dotlagCfmLtrChassisIdSubtype   LldpChassisIdSubtype,
  dotlagCfmLtrChassisId          LldpChassisId,
  dotlagCfmLtrManAddressDomain   TDomain,
  dotlagCfmLtrManAddress         TAddress,
  dotlagCfmLtrIngress            DotlagCfmIngressActionFieldValue,
  dotlagCfmLtrIngressMac         MacAddress,
  dotlagCfmLtrIngressPortIdSubtype LldpPortIdSubtype,
  dotlagCfmLtrIngressPortId     LldpPortId,
  dotlagCfmLtrEgress              DotlagCfmEgressActionFieldValue,
  dotlagCfmLtrEgressMac          MacAddress,
  dotlagCfmLtrEgressPortIdSubtype LldpPortIdSubtype,
  dotlagCfmLtrEgressPortId       LldpPortId,
  dotlagCfmLtrOrganizationSpecificTlv OCTET STRING
}

dotlagCfmLtrSeqNumber 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
    "12.14.7.5.2:b"
 ::= { dotlagCfmLtrEntry 1}

dotlagCfmLtrReceiveOrder 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.  dotlagCfmLtrReceiveOrder

```

```
    are assigned sequentially from 1, in the order that the
    Linktrace Initiator received the LTRs.

    "
    REFERENCE
        "12.14.7.5.2:c"
        ::= { dotlagCfmLtrEntry 2 }

dot1agCfmLtrTtl OBJECT-TYPE
    SYNTAX      Unsigned32 (0..255)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "TTL field value for a returned LTR."
    REFERENCE
        "12.14.7.5 and 20.36.2.2"
        ::= { dotlagCfmLtrEntry 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"
        ::= { dotlagCfmLtrEntry 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"
        ::= { dotlagCfmLtrEntry 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"
        ::= { dotlagCfmLtrEntry 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"
 ::= { dotlagCfmLtrEntry 7 }

dotlagCfmLtrRelay OBJECT-TYPE
  SYNTAX      DotlagCfmRelayActionFieldValue
  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"
 ::= { dotlagCfmLtrEntry 8 }

dotlagCfmLtrChassisIdSubtype 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 dotlagCfmLtrChassisId has a length of 0."
REFERENCE
  "802.1ag clauses 12.14.7.5.3:h and 21.5.3.2"
 ::= { dotlagCfmLtrEntry 9 }

dotlagCfmLtrChassisId 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 dotlagCfmLtrChassisIdSubtype object.
"
REFERENCE
  "802.1ag clauses 12.14.7.5.3:i and 21.5.3.3"
 ::= { dotlagCfmLtrEntry 10 }

dotlagCfmLtrManAddressDomain 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 }

```

dotlagCfmLtrIngressPortIdSubtype OBJECT-TYPE
  SYNTAX      LldpPortIdSubtype
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Format of the Ingress Port ID.
     If the dotlagCfmLtrIngress 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"
  ::= { dotlagCfmLtrEntry 15 }

dotlagCfmLtrIngressPortId 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 dotlagCfmLtrIngressPortIdSubtype object.
     If the dotlagCfmLtrIngress 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"
  ::= { dotlagCfmLtrEntry 16 }

dotlagCfmLtrEgress OBJECT-TYPE
  SYNTAX      DotlagCfmEgressActionFieldValue
  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"
  ::= { dotlagCfmLtrEntry 17 }

dotlagCfmLtrEgressMac OBJECT-TYPE
  SYNTAX      MacAddress
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "MAC address returned in the egress MAC address field.
     If the dotlagCfmLtrEgress 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"
  ::= { dotlagCfmLtrEntry 18 }

dotlagCfmLtrEgressPortIdSubtype OBJECT-TYPE
  SYNTAX      LldpPortIdSubtype
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Format of the egress Port ID.
     If the dotlagCfmLtrEgress 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"
 ::= { dotlagCfmLtrEntry 19 }

dotlagCfmLtrEgressPortId 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 dotlagCfmLtrEgressPortIdSubtype object.
     If the dotlagCfmLtrEgress 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"
 ::= { dotlagCfmLtrEntry 20 }

dotlagCfmLtrOrganizationSpecificTlv 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"
 ::= { dotlagCfmLtrEntry 21 }

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

dotlagCfmMepDbTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF DotlagCfmMepDbEntry
  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
    "19.2.15"
 ::= { dotlagCfmMep 3 }

dotlagCfmMepDbEntry OBJECT-TYPE
  SYNTAX      DotlagCfmMepDbEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The MEP Database table entry."
  INDEX { dotlagCfmMdIndex,
          dotlagCfmMaIndex,
          dotlagCfmMepIdentifier,
          dotlagCfmMepDbRMepIdentifier }
```

```

        }
 ::= { dotlagCfmMepDbTable 1 }

DotlagCfmMepDbEntry ::= SEQUENCE {
    dotlagCfmMepDbRMepIdentifier          DotlagCfmMepId,
    dotlagCfmMepDbRMepState               DotlagCfmRemoteMepState,
    dotlagCfmMepDbRMepFailedOkTime       TimeStamp,
    dotlagCfmMepDbMacAddress             MacAddress,
    dotlagCfmMepDbRdi                   TruthValue,
    dotlagCfmMepDbPortStatusTlv         DotlagCfmPortStatus,
    dotlagCfmMepDbInterfaceStatusTlv   DotlagCfmInterfaceStatus,
    dotlagCfmMepDbChassisIdSubtype     LldpChassisIdSubtype,
    dotlagCfmMepDbChassisId            LldpChassisId,
    dotlagCfmMepDbManAddressDomain     TDomain,
    dotlagCfmMepDbManAddress           TAddress,
    dotlagCfmMepDbRMepIsActive         TruthValue
}

dotlagCfmMepDbRMepIdentifier OBJECT-TYPE
    SYNTAX      DotlagCfmMepId
    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
        "12.14.7.6.2:b"
 ::= { dotlagCfmMepDbEntry 1 }

dotlagCfmMepDbRMepState OBJECT-TYPE
    SYNTAX      DotlagCfmRemoteMepState
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operational state of the remote MEP IFF State machines."
    REFERENCE
        "12.14.7.6.3:b and 20.22"
 ::= { dotlagCfmMepDbEntry 2 }

dotlagCfmMepDbRMepFailedOkTime 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
        "12.14.7.6.3:c"
 ::= { dotlagCfmMepDbEntry 3 }

dotlagCfmMepDbMacAddress OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The MAC address of the remote MEP."
    REFERENCE

```

```
        "12.14.7.6.3:d and 20.19.7"
        ::= { dotlagCfmMepDbEntry 4 }

dotlagCfmMepDbRdi 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"
        ::= { dotlagCfmMepDbEntry 5 }

dotlagCfmMepDbPortStatusTlv OBJECT-TYPE
    SYNTAX      DotlagCfmPortStatus
    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 no port status TLV was received in the last CCM.
         "
    REFERENCE
        "12.14.7.6.3:f and 20.19.3"
    DEFVAL { psNoPortStateTLV }
    ::= { dotlagCfmMepDbEntry 6}

dotlagCfmMepDbInterfaceStatusTlv OBJECT-TYPE
    SYNTAX      DotlagCfmInterfaceStatus
    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
        "12.14.7.6.3:g and 20.19.4"
    DEFVAL { isNoInterfaceStatusTLV }
    ::= { dotlagCfmMepDbEntry 7}

dotlagCfmMepDbChassisIdSubtype 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"
        ::= { dotlagCfmMepDbEntry 8 }

dotlagCfmMepDbChassisId 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 dotlagCfmLtrChassisIdSubtype object.
  "
REFERENCE
  "802.1ag clauses 12.14.7.6.3:h and 21.5.3.3"
 ::= { dotlagCfmMepDbEntry 9 }

```

```

dotlagCfmMepDbManAddressDomain OBJECT-TYPE
SYNTAX      TDomain
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "The TDomain that identifies the type and format of
   the related dotlagCfmMepDbManAddress 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 dotlagCfmMepDbManAddress 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"
 ::= { dotlagCfmMepDbEntry 10 }

```

```

dotlagCfmMepDbManAddress 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 dotlagCfmMepDbManAddressDomain contains the value 'zeroDotZero', this object dotlagCfmMepDbManAddress 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"
 ::= { dotlagCfmMepDbEntry 11 }

```

```

dotlagCfmMepDbRMepIsActive OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
  "A Boolean value stating if the remote MEP is active."
REFERENCE

```

```
"12.14.7.1.3:af"
 ::= { dotlagCfmMepDbEntry 12 }

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

dotlagCfmFaultAlarm NOTIFICATION-TYPE
 OBJECTS      { dotlagCfmMepHighestPrDefect
                  }
 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 dotlagCfmMepHighestPrDefect
variable in the notification:

dotlagCfmMdIndex - Also the index of the MEP's
                   Maintenance Domain table entry
                   (dotlagCfmMdTable).
dotlagCfmMaIndex - Also an index (with the MD table index)
                   of the MEP's Maintenance Association
                   network table entry
                   (dotlagCfmMaNetTable), and (with the MD
                   table index and component ID) of the
                   MEP's MA component table entry
                   (dotlagCfmMaCompTable).
dotlagCfmMepIdentifier - MEP Identifier and final index
                         into the MEP table (dotlagCfmMepTable).

"
REFERENCE
"12.14.7.7"
 ::= { dotlagNotifications 1 }

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

dotlagCfmCompliances OBJECT IDENTIFIER ::= { dotlagCfmConformance 1 }
dotlagCfmGroups      OBJECT IDENTIFIER ::= { dotlagCfmConformance 2 }

-- *****
```

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

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

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

dot1agCfmConfigErrorListGroup OBJECT-GROUP
    OBJECTS {
        dot1agCfmConfigErrorListErrorType
    }
    STATUS      deprecated
    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      deprecated
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,
dot1agCfmRowStatus,
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."
        ::= { dotlagCfmGroups 9 }

ieee8021CfmMaNetGroup OBJECT-GROUP
OBJECTS {
    dotlagCfmMaNetFormat,
    dotlagCfmMaNetName,
    dotlagCfmMaNetCcmInterval,
    dotlagCfmMaNetRowStatus,
    dotlagCfmMaMepListRowStatus

}
STATUS      current
DESCRIPTION
        "Objects for the MA Net group."
        ::= { dotlagCfmGroups 10 }

ieee8021CfmDefaultMdDefGroup OBJECT-GROUP
OBJECTS {
    dotlagCfmDefaultMdDefLevel,
    dotlagCfmDefaultMdDefMhfCreation,
    dotlagCfmDefaultMdDefIdPermission

}
STATUS      current
DESCRIPTION
        "Objects for the Default MD default Level group."
        ::= { dotlagCfmGroups 11 }

ieee8021CfmPbbTeExtensionGroup OBJECT-GROUP
OBJECTS {
    dotlagCfmMepDbRMepIsActive,
    dotlagCfmMepPbbTransmitLbmLtmReverseVid

}
STATUS      current
DESCRIPTION
        "Objects needed for systems that support PBB-TE CFM functionality."
        ::= { dotlagCfmGroups 12 }

ieee8021CfmPbbTeTrafficBitGroup OBJECT-GROUP
OBJECTS {
    dotlagCfmMepDbManAddress,
    dotlagCfmMepPbbTeCanReportPbbTePresence,
    dotlagCfmMepPbbTeMismatchAlarm,
    dotlagCfmMepPbbTeTrafficMismatchDefect,
    dotlagCfmMepPbbTeLocalMismatchDefect,
    dotlagCfmMepPbbTeMismatchSinceReset

}
STATUS      current
DESCRIPTION
        "Objects needed for PBB-TE supporting systems that support the
         optional traffic bit."
        ::= { dotlagCfmGroups 13 }

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

dotlagCfmCompliance MODULE-COMPLIANCE
```

```
STATUS      deprecated
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."
```

```
 ::= { dotlagCfmCompliances 1 }
END
```

17.7.7.2 Definitions for the IEEE8021-CFM-V2 MIB module

```
IEEE8021-CFM-V2-MIB DEFINITIONS ::= BEGIN

-- *****
-- IEEE 802.1ag (TM) CFM MIB
-- *****

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32          FROM SNMPv2-SMI      -- [RFC2578]
    RowStatus,
    TruthValue, MacAddress
                           FROM SNMPv2-TC      -- [RFC2579]
    MODULE-COMPLIANCE,
    OBJECT-GROUP
                           FROM SNMPv2-CONF     -- [RFC2580]
    InterfaceIndex
                           FROM IF-MIB          -- [RFC2863]
    IEEE8021ServiceSelectorType,
    IEEE8021ServiceSelectorValue,
    IEEE8021ServiceSelectorValueOrNone,
    IEEE8021PbbComponentIdentifier,
    ieee802dot1mibs       FROM IEEE8021-TC-MIB

--cfm types
    Dot1agCfmMhfCreation,
    Dot1agCfmIdPermission,
    Dot1agCfmMDLevel,
    Dot1agCfmMpDirection,
    Dot1agCfmMepIdOrZero,
    Dot1agCfmMDLevelOrNone,
    Dot1agCfmConfigErrors,
-- cfm indexes
    dot1agCfmMdIndex,
    dot1agCfmMaIndex,
--cfm groups
    dot1agCfmStack,
    dot1agCfmDefaultMd,
    dot1agCfmVlan,
    dot1agCfmConfigErrorList,
    dot1agCfmMa,
-- cfm row items
    dot1agCfmMepLbrBadMsdu,
    dot1agCfmMdRowStatus,
    dot1agCfmMaNetRowStatus,
    dot1agCfmMaMepListRowStatus,
    dot1agCfmMepRowStatus,
--cfm conformance groups
    dot1agCfmCompliances,
    dot1agCfmGroups,
    dot1agCfmMdGroup,
    dot1agCfmMepGroup,
    dot1agCfmMepDbGroup,
    dot1agCfmNotificationsGroup,
    ieee8021CfmDefaultMdDefGroup,
    ieee8021CfmMaNetGroup,
    ieee8021CfmPbbTeExtensionGroup,
```

ieee8021CfmPbbTeTrafficBitGroup FROM IEEE8021-CFM-MIB
;

ieee8021CfmV2Mib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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
Postal: C/O IEEE 802.1 Working Group
IEEE Standards Association
445 Hoes Lane
P.O. Box 1331
Piscataway
NJ 08855-1331
USA
E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG

Contact: Norman Finn
Postal: C/O IEEE 802.1 Working Group
IEEE Standards Association
445 Hoes Lane
P.O. Box 1331
Piscataway
NJ 08855-1331
USA
E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG

Contact: David Levi
Postal: C/O IEEE 802.1 Working Group
IEEE Standards Association
445 Hoes Lane
P.O. Box 1331
Piscataway
NJ 08855-1331
USA
E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG
"

DESCRIPTION
"Connectivity Fault Management V2 module.

Unless otherwise indicated, the references in this MIB module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."

REVISION "201102270000Z" -- February 27, 2011
DESCRIPTION
"Minor edits to contact information etc. as part of 2011 revision of IEEE Std 802.1Q."

REVISION "200811180000Z" -- November 18, 2008
DESCRIPTION
"Added a new compliance clause for utilization by systems that support CFM and PBB-TE."

```

REVISION      "200810150000Z" -- October 15, 2008
DESCRIPTION
  "The IEEE8021-CFM-V2-MIB Module contains objects that
   replace those deprecated in the IEEE8021-CFM-MIB module.

  This version is included in IEEE 802.1ap"

 ::= { ieee802dot1mibs 7 }

-- ****
-- Note: Re-indexed 802.1ag tables
-- ****
-- This section contains new tables replacing deprecated tables in
-- this version of the MIB
-- ****
-- The CFM Stack Table
-- ****

ieee8021CfmStackTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF Ieee8021CfmStackEntry
  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
    "12.14.2"
 ::= { dotlagCfmStack 2 }

ieee8021CfmStackEntry OBJECT-TYPE
  SYNTAX      Ieee8021CfmStackEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The Stack table entry"
  INDEX { ieee8021CfmStackifIndex, ieee8021CfmStackServiceSelectorType,
          ieee8021CfmStackServiceSelectorOrNone,
          ieee8021CfmStackMdLevel, ieee8021CfmStackDirection
        }
 ::= { ieee8021CfmStackTable 1 }

Ieee8021CfmStackEntry ::= SEQUENCE {
  ieee8021CfmStackifIndex                      InterfaceIndex,
  ieee8021CfmStackServiceSelectorType           IEEE8021ServiceSelectorType,
  ieee8021CfmStackServiceSelectorOrNone         ieee8021CfmStackServiceSelectorOrNone
  IEEE8021ServiceSelectorValueOrNone,
  ieee8021CfmStackMdLevel                      Dot1agCfmMDLevel,
  ieee8021CfmStackDirection                    Dot1agCfmMpDirection,
  ieee8021CfmStackMdIndex                     Unsigned32,
  ieee8021CfmStackMaIndex                     Unsigned32,
  ieee8021CfmStackMepId                      Dot1agCfmMepIdOrZero,
  ieee8021CfmStackMacAddress                  MacAddress
}

```

```
ieee8021CfmStackifIndex 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
        ieee8021CfmStackTable, 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
        ieee8021CfmStackTable with the interface index.

        "
    REFERENCE
        "12.14.2.1.2:a"
    ::= { ieee8021CfmStackEntry 1 }

ieee8021CfmStackServiceSelectorType OBJECT-TYPE
    SYNTAX      IEEE8021ServiceSelectorType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Type of the Service Selector identifier indicated by
        ieee8021CfmStackServiceSelectorOrNone.
        See textual convention IEEE8021ServiceSelectorType for details.

        "
    REFERENCE
        "12.14.2.1.2:d, 22.1.7"
    ::= { ieee8021CfmStackEntry 2 }

ieee8021CfmStackServiceSelectorOrNone OBJECT-TYPE
    SYNTAX      IEEE8021ServiceSelectorValueOrNone
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Service Selector identifier to which the MP is attached, or 0, if none.
        See textual convention IEEE8021ServiceSelectorValue for details.

        "
    REFERENCE
        "12.14.2.1.2:d, 22.1.7"
    ::= { ieee8021CfmStackEntry 3 }

ieee8021CfmStackMdLevel OBJECT-TYPE
    SYNTAX      Dot1agCfmMDLevel
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "MD Level of the Maintenance Point."
    REFERENCE
        "12.14.2.1.2:b"
    ::= { ieee8021CfmStackEntry 4 }

ieee8021CfmStackDirection OBJECT-TYPE
    SYNTAX      Dot1agCfmMpDirection
    MAX-ACCESS  not-accessible
    STATUS      current
```

```

DESCRIPTION
    "Direction in which the MP faces on the Bridge Port"
REFERENCE
    "12.14.2.1.2:c"
 ::= { ieee8021CfmStackEntry 5 }

ieee8021CfmStackMdIndex OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The index of the Maintenance Domain in the ieee8021CfmMdTable
     to which the MP is associated, or 0, if none."
REFERENCE
    "12.14.2.1.3:b"
 ::= { ieee8021CfmStackEntry 6 }

ieee8021CfmStackMaIndex OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The index of the MA in the ieee8021CfmMaNetTable and
     ieee8021CfmMaCompTable to which the MP is associated, or 0, if
     none."
REFERENCE
    "12.14.2.1.3:c"
 ::= { ieee8021CfmStackEntry 7 }

ieee8021CfmStackMepId OBJECT-TYPE
SYNTAX      Dot1agCfmMepIdOrZero
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "If an MEP is configured, the MEPID, else 0"
REFERENCE
    "12.14.2.1.3:d"
 ::= { ieee8021CfmStackEntry 8 }

ieee8021CfmStackMacAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "MAC address of the MP."
REFERENCE
    "12.14.2.1.3:e"
 ::= { ieee8021CfmStackEntry 9 }

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

ieee8021CfmVlanTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021CfmVlanEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "This table defines the association of VIDs into VLANs. There
     are two entries per VLAN, one for the upstream direction and one
     for the downstream direction. The upstream entry contains the
     VID and the downstream entry contains the corresponding
     priority level."
```

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

"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."
 ::= { dotlagCfmVlan 2 }

ieee8021CfmVlanEntry OBJECT-TYPE
SYNTAX Ieee8021CfmVlanEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The VLAN table entry."
INDEX { ieee8021CfmVlanComponentId,
ieee8021CfmVlanSelector}
 ::= { ieee8021CfmVlanTable 1 }

Ieee8021CfmVlanEntry ::= SEQUENCE {
ieee8021CfmVlanComponentId IEEE8021PbbComponentIdentifier,
ieee8021CfmVlanSelector IEEE8021ServiceSelectorValue,
ieee8021CfmVlanPrimarySelector IEEE8021ServiceSelectorValue,
ieee8021CfmVlanRowStatus RowStatus
}

ieee8021CfmVlanComponentId OBJECT-TYPE
SYNTAX IEEE8021PbbComponentIdentifier
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The bridge component within the system to which the information in this ieee8021CfmVlanEntry 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

"12.3 1)"

::= { ieee8021CfmVlanEntry 1 }

ieee8021CfmVlanSelector OBJECT-TYPE
SYNTAX IEEE8021ServiceSelectorValue
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"This is a service ID belonging to a service that is associated

with more than one Service Selector identifiers, and this is not the Primary Service ID of the service. The type of this Service Selector is the same as the primary Service Selector's type defined by ieee8021CfmMaCompPrimarySelectorType in the ieee8021CfmMaCompTable.

"

::= { ieee8021CfmVlanEntry 3 }

ieee8021CfmVlanPrimarySelector OBJECT-TYPE

SYNTAX IEEE8021ServiceSelectorValue

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This is the Primary Service selector for a Service that is associated with more than one Service Selector identifiers. This value MUST not equal the value of ieee8021CfmVlanSelector. The type of this Service Selector is the same as the primary Service Selector's type defined by ieee8021CfmMaCompPrimarySelectorType in the ieee8021CfmMaCompTable.

"

::= { ieee8021CfmVlanEntry 5 }

ieee8021CfmVlanRowStatus 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.

"

::= { ieee8021CfmVlanEntry 6 }

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

ieee8021CfmDefaultMdTable OBJECT-TYPE

SYNTAX SEQUENCE OF Ieee8021CfmDefaultMdEntry

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 ieee8021CfmDefaultMdStatus 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

"12.14.3"

::= { dot1agCfmDefaultMd 5 }

ieee8021CfmDefaultMdEntry OBJECT-TYPE

SYNTAX Ieee8021CfmDefaultMdEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The Default MD Level table entry."

INDEX { ieee8021CfmDefaultMdComponentId,
ieee8021CfmDefaultMdPrimarySelectorType,
ieee8021CfmDefaultMdPrimarySelector}

::= { ieee8021CfmDefaultMdTable 1 }

Ieee8021CfmDefaultMdEntry ::= SEQUENCE {

ieee8021CfmDefaultMdComponentId	IEEE8021PbbComponentIdentifier,
ieee8021CfmDefaultMdPrimarySelectorType	IEEE8021ServiceSelectorType,
ieee8021CfmDefaultMdPrimarySelector	IEEE8021ServiceSelectorValue,
ieee8021CfmDefaultMdStatus	TruthValue,
ieee8021CfmDefaultMdLevel	Dot1agCfmMDLevelOrNone,
ieee8021CfmDefaultMdMhfCreation	Dot1agCfmMhfCreation,
ieee8021CfmDefaultMdIdPermission	Dot1agCfmIdPermission

}

ieee8021CfmDefaultMdComponentId OBJECT-TYPE

SYNTAX IEEE8021PbbComponentIdentifier

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The bridge component within the system to which the information in this ieee8021CfmDefaultMdEntry 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

"12.3 1)"

::= { ieee8021CfmDefaultMdEntry 1 }

ieee8021CfmDefaultMdPrimarySelectorType OBJECT-TYPE

SYNTAX IEEE8021ServiceSelectorType

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Type of the Primary Service Selector identifier indicated by ieee8021CfmDefaultMdPrimarySelector. See textual convention IEEE8021ServiceSelectorType for details."

```

"
 ::= { ieee8021CfmDefaultMdEntry 2 }

ieee8021CfmDefaultMdPrimarySelector OBJECT-TYPE
    SYNTAX      IEEE8021ServiceSelectorValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Primary Service Selector identifier of a Service Instance with
         no MA configured. See IEEE8021ServiceSelectorValue for details.
"
 ::= { ieee8021CfmDefaultMdEntry 3 }

ieee8021CfmDefaultMdStatus 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
    "12.14.3.1.3:b"
 ::= { ieee8021CfmDefaultMdEntry 4 }

ieee8021CfmDefaultMdLevel 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
         ieee8021CfmDefaultMdDefLevel.
"
REFERENCE
    "12.14.3.1.3:c, 12.14.3.2.2:b"
DEFVAL {-1}
 ::= { ieee8021CfmDefaultMdEntry 5 }

ieee8021CfmDefaultMdMhfCreation 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 ieee8021CfmDefaultMdDefMhfCreation.

        The value of this variable is meaningless if the values of
        ieee8021CfmDefaultMdStatus is false.
"
REFERENCE
    "12.14.3.1.3:d"

```

```
DEFVAL {defMHFdefer}
 ::= { ieee8021CfmDefaultMdEntry 6 }

ieee8021CfmDefaultMdIdPermission OBJECT-TYPE
    SYNTAX      DotlagCfmIdPermission
    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 ieee8021CfmDefaultMdDefIdPermission.

        The value of this variable is meaningless if the values of
        ieee8021CfmDefaultMdStatus is false.
    "
    REFERENCE
        "12.14.3.1.3:e"
    DEFVAL { sendIdDefer }
    ::= { ieee8021CfmDefaultMdEntry 7 }

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

ieee8021CfmConfigErrorListTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021CfmConfigErrorListEntry
    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
        "12.14.4"
    ::= { dotlagCfmConfigErrorList 2}

ieee8021CfmConfigErrorListEntry OBJECT-TYPE
    SYNTAX      Ieee8021CfmConfigErrorListEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Config Error List Table entry"
    INDEX { ieee8021CfmConfigErrorListSelectorType,
            ieee8021CfmConfigErrorListSelector,
            ieee8021CfmConfigErrorListIfIndex
        }
    ::= { ieee8021CfmConfigErrorListTable 1}

Ieee8021CfmConfigErrorListEntry ::= SEQUENCE {
                                ieee8021CfmConfigErrorListSelectorType
                                IEEE8021ServiceSelectorType,
                                ieee8021CfmConfigErrorListSelector
                                IEEE8021ServiceSelectorValue,
                                ieee8021CfmConfigErrorListIfIndex
                                                InterfaceIndex,
                                ieee8021CfmConfigErrorListErrorType
                                                DotlagCfmConfigErrors
                            }
```

```
ieee8021CfmConfigErrorListSelectorType OBJECT-TYPE
    SYNTAX      IEEE8021ServiceSelectorType
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Type of the Service Selector identifier indicated by
         ieee8021CfmConfigErrorListSelector. See textual
         convention IEEE8021ServiceSelectorType for details.
        "
    REFERENCE
        "12.14.4.1.2:a"
    ::= { ieee8021CfmConfigErrorListEntry 1 }

ieee8021CfmConfigErrorListSelector OBJECT-TYPE
    SYNTAX      IEEE8021ServiceSelectorValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Service Selector Identifier of the Service with interfaces
         in error. See IEEE8021ServiceSelectorValue for details.
        "
    REFERENCE
        "12.14.4.1.2:a"
    ::= { ieee8021CfmConfigErrorListEntry 2 }

ieee8021CfmConfigErrorListIfIndex 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
ieee8021CfmConfigErrorListTable indexed by that
InterfaceIndex value.
"
    REFERENCE
        "12.14.4.1.2:b"
    ::= { ieee8021CfmConfigErrorListEntry 3 }

ieee8021CfmConfigErrorListErrorType 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
```

```
    "12.14.4.1.3:b"
 ::= { ieee8021CfmConfigErrorListEntry 4 }

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

ieee8021CfmMaCompTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021CfmMaCompEntry
    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 ieee8021CfmMaNetTable.

This table uses three indices, first index is the IEEE8021PbbComponentIdentifier that identifies the component within the Bridge for which the information in the ieee8021CfmMaCompEntry applies. The second is the index of the Maintenance Domain table. The third index is the same as the index of the ieee8021CfmMaNetEntry for the same MA.

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

```
""
REFERENCE
"18.2"
 ::= { dotlagCfmMa 4 }

ieee8021CfmMaCompEntry OBJECT-TYPE
    SYNTAX      Ieee8021CfmMaCompEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The MA table entry."
INDEX {ieee8021CfmMaComponentId,
       dotlagCfmMdIndex, dotlagCfmMaIndex }
 ::= { ieee8021CfmMaCompTable 1 }

Ieee8021CfmMaCompEntry ::= SEQUENCE {
    ieee8021CfmMaComponentId          IEEE8021PbbComponentIdentifier,
    ieee8021CfmMaCompPrimarySelectorType IEEE8021ServiceSelectorType,
                                         ieee8021CfmMaCompPrimarySelectorOrNone
    IEEE8021ServiceSelectorValueOrNone,
    ieee8021CfmMaCompMhfCreation       Dot1agCfmMhfCreation,
    ieee8021CfmMaCompIdPermission      Dot1agCfmIdPermission,
    ieee8021CfmMaCompNumberOfVids     Unsigned32,
    ieee8021CfmMaCompRowStatus        RowStatus
}

ieee8021CfmMaComponentId OBJECT-TYPE
```

```

SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
  "The bridge component within the system to which the information
   in this ieee8021CfmMaCompEntry 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
  "12.3 1"
 ::= { ieee8021CfmMaCompEntry 1 }

ieee8021CfmMaCompPrimarySelectorType OBJECT-TYPE
SYNTAX      IEEE8021ServiceSelectorType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "Type of the Service Selector identifiers indicated by
   ieee8021CfmMaCompPrimarySelectorOrNone. If the service instance is
defined by more
than one Service Selector, this parameter also indicates the type of the
  ieee8021CfmVlanPrimarySelector and ieee8021CfmVlanSelector in the
ieee8021CfmVlanTable.
  In Services instances made of multiple Service Selector identifiers,
ensures that the
    type of the Service selector identifiers is the same. See textual
convention
    DotlagCfmServiceSelectorType for details.
"
REFERENCE
  "12.14.6.1.3:b"
 ::= { ieee8021CfmMaCompEntry 2 }

ieee8021CfmMaCompPrimarySelectorOrNone OBJECT-TYPE
SYNTAX      IEEE8021ServiceSelectorValueOrNone
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "Service Selector identifier to which the MP is attached, or 0, if none.
  If the MA is associated with more than one Service Selectors Identifiers,
the
  ieee8021CfmVlanTable lists them.
"
REFERENCE
  "12.14.6.1.3:b"
 ::= { ieee8021CfmMaCompEntry 3 }

ieee8021CfmMaCompMhfCreation OBJECT-TYPE
SYNTAX      DotlagCfmMhfCreation
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "Indicates if the Management entity can create MHFs (MIP Half
   Function) for this MA.
"
REFERENCE
  "12.14.6.1.3:c"

```

```
DEFVAL { defMHFdefer }
 ::= { ieee8021CfmMaCompEntry 4 }

ieee8021CfmMaCompIdPermission OBJECT-TYPE
    SYNTAX      DotlagCfmIdPermission
    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
        "12.14.6.1.3:d"
    DEFVAL { sendIdDefer }
 ::= { ieee8021CfmMaCompEntry 5 }

ieee8021CfmMaCompNumberOfVids OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The number of VIDs associated with the MA.
        "
    REFERENCE
        "12.14.6.1.3:b"
 ::= { ieee8021CfmMaCompEntry 6 }

ieee8021CfmMaCompRowStatus 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.
        "
 ::= { ieee8021CfmMaCompEntry 7 }

-- *****
-- Units of conformance
-- *****

ieee8021CfmStackGroup OBJECT-GROUP
    OBJECTS {
        ieee8021CfmStackMdIndex,
        ieee8021CfmStackMaIndex,
        ieee8021CfmStackMepId,
        ieee8021CfmStackMacAddress
    }
    STATUS      current
    DESCRIPTION
        "Objects for the Stack group."
 ::= { dotlagCfmGroups 12 }

ieee8021CfmMaGroup OBJECT-GROUP
    OBJECTS {
```

```
ieee8021CfmMaCompPrimarySelectorType,
ieee8021CfmMaCompPrimarySelectorOrNone,
ieee8021CfmMaCompMhfCreation,
ieee8021CfmMaCompIdPermission,
ieee8021CfmMaCompRowStatus,
ieee8021CfmMaCompNumberOfVids
}
STATUS      current
DESCRIPTION
    "Objects for the MA group."
 ::= { dotlagCfmGroups 13 }

ieee8021CfmDefaultMdGroup OBJECT-GROUP
OBJECTS {
    ieee8021CfmDefaultMdStatus,
    ieee8021CfmDefaultMdLevel,
    ieee8021CfmDefaultMdMhfCreation,
    ieee8021CfmDefaultMdIdPermission
}
STATUS      current
DESCRIPTION
    "Objects for the Default MD Level group."
 ::= { dotlagCfmGroups 14 }

ieee8021CfmVlanIdGroup OBJECT-GROUP
OBJECTS {
    ieee8021CfmVlanPrimarySelector,
    ieee8021CfmVlanRowStatus
}
STATUS      current
DESCRIPTION
    "Objects for the VLAN ID group."
 ::= { dotlagCfmGroups 15 }

ieee8021CfmConfigErrorListGroup OBJECT-GROUP
OBJECTS {
    ieee8021CfmConfigErrorListErrorType
}
STATUS      current
DESCRIPTION
    "Objects for the CFM Configuration Error List Group."
 ::= {dotlagCfmGroups 16 }

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

ieee8021CfmComplianceV2 MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for support of the CFM MIB module."

MODULE
MANDATORY-GROUPS {
    ieee8021CfmStackGroup,
    ieee8021CfmMaGroup,
    ieee8021CfmDefaultMdGroup,
    ieee8021CfmConfigErrorListGroup
}
```

```
GROUP ieee8021CfmVlanIdGroup
DESCRIPTION "The VLAN ID group is optional."

OBJECT ieee8021CfmMaCompRowStatus
SYNTAX      RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                        destroy(6) }
DESCRIPTION "Support for createAndWait is not required."

OBJECT ieee8021CfmVlanRowStatus
SYNTAX      RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                        destroy(6) }
DESCRIPTION "Support for createAndWait is not required."

MODULE IEEE8021-CFM-MIB
MANDATORY-GROUPS {
    dot1agCfmMdGroup,
    dot1agCfmMepGroup,
    dot1agCfmMepDbGroup,
    dot1agCfmNotificationsGroup,
    ieee8021CfmDefaultMdDefGroup,
    ieee8021CfmMaNetGroup
}

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 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 2 }

dot1agCfmWithPbbTeCompliance MODULE-COMPLIANCE
```

```

STATUS      current
DESCRIPTION
  "The compliance statement for support of the CFM MIB for
  systems that support PBB-TE."

MODULE
  MANDATORY-GROUPS {
    ieee8021CfmStackGroup,
    ieee8021CfmMaGroup,
    ieee8021CfmDefaultMdGroup,
    ieee8021CfmConfigErrorListGroup
  }

GROUP ieee8021CfmVlanIdGroup
DESCRIPTION "The VLAN ID group is optional."

OBJECT ieee8021CfmMaCompRowStatus
  SYNTAX      RowStatus { active(1), notInService(2) }
  WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                           destroy(6) }
  DESCRIPTION "Support for createAndWait is not required."

OBJECT ieee8021CfmVlanRowStatus
  SYNTAX      RowStatus { active(1), notInService(2) }
  WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                           destroy(6) }
  DESCRIPTION "Support for createAndWait is not required."

MODULE IEEE8021-CFM-MIB
  MANDATORY-GROUPS {
    dot1agCfmMdGroup,
    dot1agCfmMepGroup,
    dot1agCfmMepDbGroup,
    dot1agCfmNotificationsGroup,
    ieee8021CfmDefaultMdDefGroup,
    ieee8021CfmMaNetGroup,
    ieee8021CfmPbbTeExtensionGroup
  }

GROUP ieee8021CfmPbbTeTrafficBitGroup
DESCRIPTION "The objects needed to support the traffic bit are optional
as traffic bit support, itself, 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  dotlagCfmMaMepListRowStatus  
SYNTAX    RowStatus { active(1), notInService(2) }  
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),  
                      destroy(6) }  
DESCRIPTION "Support for createAndWait is not required."  
  
OBJECT  dotlagCfmMepRowStatus  
SYNTAX    RowStatus { active(1), notInService(2) }  
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),  
                      destroy(6) }  
DESCRIPTION "Support for createAndWait is not required."  
  
 ::= { dotlagCfmCompliances 3 }  
  
END
```

17.7.8 Definitions for the IEEE8021-PBB MIB module

```

IEEE8021-PBB-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Unsigned32, Integer32
        FROM SNMPv2-SMI
    RowStatus, StorageType, MacAddress, TruthValue
        FROM SNMPv2-TC
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    ieee802dot1mibs, IEEE8021PbbComponentIdentifier,
    IEEE8021BridgePortNumber, IEEE8021PbbServiceIdentifier,
    IEEE8021PbbServiceIdentifierOrUnassigned, IEEE8021PbbIngressEgress,
    IEEE8021PriorityValue, IEEE8021PriorityCodePoint
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBasePortComponentId, ieee8021BridgeBasePort
        FROM IEEE8021-BRIDGE-MIB
    VlanId
        FROM Q-BRIDGE-MIB
    InterfaceIndex, InterfaceIndexOrZero
        FROM IF-MIB
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF;

ieee8021PbbMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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 Levi
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
"The Provider Backbone Bridge (PBB) MIB module for managing
devices that support PBB.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
REVISION      "201102270000Z" -- February 27, 2011
DESCRIPTION
"Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."
REVISION      "200811180000Z" -- November 18, 2008

```

```
DESCRIPTION
    "Modified VIP table to support the enableConnectionIdentifier
     configuration option."
REVISION      "200810150000Z" -- October 15, 2008
DESCRIPTION
    "Initial version published in 802.1ap."
 ::= { ieee802dot1mibs 9 }

ieee8021PbbNotifications   OBJECT IDENTIFIER ::= { ieee8021PbbMib 0 }
ieee8021PbbObjects         OBJECT IDENTIFIER ::= { ieee8021PbbMib 1 }
ieee8021PbbConformance    OBJECT IDENTIFIER ::= { ieee8021PbbMib 2 }

-- 
-- 802.1ah MIB Objects
--

ieee8021PbbProviderBackboneBridge
OBJECT IDENTIFIER ::= { ieee8021PbbObjects 1 }

-- -----
-- 12.16.1.1/2 Backbone Edge Bridge (BEB) configuration
-- -----
-- items a), b), c), e), and g), see below
-- d) ieee8021BridgeBaseBridgeAddress from IEEE8021-BRIDGE-MIB
-- f) ieee8021BridgeBaseBridgeAddress from IEEE8021-BRIDGE-MIB
-- i) and j) ifPhysAddress from the IF-MIB, the correct instance
--           can be found using ieee8021BridgeBasePortIfIndex
--           from the IEEE8021-BRIDGE-MIB
-- -----
ieee8021PbbBackboneEdgeBridgeObjects
OBJECT IDENTIFIER ::= { ieee8021PbbProviderBackboneBridge 1 }

ieee8021PbbBackboneEdgeBridgeAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The MAC Address used by the BEB when it must be referred
     to in a unique fashion."
REFERENCE   "12.16.1.1.3 a)"
 ::= { ieee8021PbbBackboneEdgeBridgeObjects 1 }

ieee8021PbbBackboneEdgeBridgeName OBJECT-TYPE
SYNTAX      SnmpAdminString (SIZE (0..32))
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "A text string of locally determined significance. This value
     must be persistent over power up restart/reboot."
REFERENCE   "12.16.1.1.3 b), 12.16.1.2.2"
 ::= { ieee8021PbbBackboneEdgeBridgeObjects 2 }

ieee8021PbbNumberOfIComponents OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
```

```

        "The number of I-components in this BEB."
REFERENCE    "12.16.1.1.3 c)"
::= { ieee8021PbbBackboneEdgeBridgeObjects 3 }

ieee8021PbbNumberOfBComponents OBJECT-TYPE
SYNTAX      Unsigned32 (0..1)
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "The number of B-components in this BEB."
REFERENCE    "12.16.1.1.3 e)"
::= { ieee8021PbbBackboneEdgeBridgeObjects 4 }

ieee8021PbbNumberOfBebPorts OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "The number of CNPs, PIPs, CBPs, and PNPs in this BEB."
REFERENCE    "12.16.1.1.3 g)"
::= { ieee8021PbbBackboneEdgeBridgeObjects 5 }

ieee8021PbbNextAvailablePipIfIndex OBJECT-TYPE
SYNTAX      InterfaceIndex
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "This object provides an available ifIndex value that can
        be used for creation of a PIP."
REFERENCE    "12.16.4.1/2"
::= { ieee8021PbbBackboneEdgeBridgeObjects 6 }

-- =====
-- 12.16.3.1/2 Virtual Instance Port (VIP) configuration table
-- =====

ieee8021PbbVipTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbbVipEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
        "This table lists the additional PBB parameters for each
        VIP. Entries in this table must be persistent over power
        up/restart/reboot."
REFERENCE    "12.16.3.1/2"
::= { ieee8021PbbProviderBackboneBridge 2 }

ieee8021PbbVipEntry OBJECT-TYPE
SYNTAX      Ieee8021PbbVipEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
        "Each entry consists of the name string, I-SID, Default
        Destination MAC, Service Type, and possible B-MAC."
INDEX       { ieee8021BridgeBasePortComponentId,
              ieee8021BridgeBasePort }
::= { ieee8021PbbVipTable 1 }

Ieee8021PbbVipEntry ::=
```

```
SEQUENCE {
    ieee8021PbbVipPipIfIndex
        InterfaceIndexOrZero,
    ieee8021PbbVipISid
        IEEE8021PbbServiceIdentifierOrUnassigned,
    ieee8021PbbVipDefaultDstBMAC
        MacAddress,
    ieee8021PbbVipType
        IEEE8021PbbIngressEgress,
    ieee8021PbbVipRowStatus
        RowStatus,
    ieee8021PbbVipEnableConnectionId
        TruthValue
}

ieee8021PbbVipPipIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndexOrZero
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Identifies the PIP associated with this VIP within the BEB.
         A value of zero indicates the VIP is not currently associated
         with any PIP.

        The value of this object must be persistent across
        reinitializations of the management system."
    DEFVAL      { 0 }
    ::= { ieee8021PbbVipEntry 1 }

ieee8021PbbVipISid OBJECT-TYPE
    SYNTAX      IEEE8021PbbServiceIdentifierOrUnassigned
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The I-SID for this VIP.

        Within an I-Component, an VIP can only be serviced
        by one I-SID. And the ISID is a configurable parameter
        of the VIP.

        The value of this object must be persistent across
        reinitializations of the management system."
    DEFVAL      { 1 }
    ::= { ieee8021PbbVipEntry 2 }

ieee8021PbbVipDefaultDstBMAC OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Default Destination B-MAC for this VIP."
    DEFVAL      { '001e83000001'h }
    ::= { ieee8021PbbVipEntry 3 }

ieee8021PbbVipType OBJECT-TYPE
    SYNTAX      IEEE8021PbbIngressEgress
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
```

"This feature is used to support asymmetric VLANs.

The value of this object must be persistent across reinitializations of the management system."

```
DEFVAL { { egress, ingress } }
 ::= { ieee8021PbbVipEntry 4 }
```

`ieee8021PbbVipRowStatus` OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This indicates the status of an entry in this table, and is used to create/delete entries.

It is an implementation specific decision as to whether any column in this table may be set while the corresponding instance of this object is valid(1).

The value of this object must be persistent across reinitializations of the management system."

```
 ::= { ieee8021PbbVipEntry 5 }
```

`ieee8021PbbVipEnableConnectionId` OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This indicates if the connection_identifier parameter is allowed to learn associations between a backbone MAC address and a customer MAC address. The default value is true, indicating that such learning is allowed. This parameter should be configured to false at the root node of a Point-to-multipoint TE service instance."

DEFVAL { true }

```
 ::= { ieee8021PbbVipEntry 6 }
```

-- =====

-- 12.16.3.1/2 I-SID to VIP cross-reference table

-- =====

`ieee8021PbbISidToVipTable` OBJECT-TYPE

SYNTAX SEQUENCE OF `Ieee8021PbbISidToVipEntry`

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table contains a cross-reference of I-SID values to the VIPs with which they are associated. This allows VIPs to be located easily by their associated I-SID."

REFERENCE "12.16.3.1/2"

```
 ::= { ieee8021PbbProviderBackboneBridge 3 }
```

`ieee8021PbbISidToVipEntry` OBJECT-TYPE

SYNTAX `Ieee8021PbbISidToVipEntry`

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"A mapping from an I-SID to the VIP with which it is associated. An entry will exist for each entry in the `ieee8021PbbVipTable`."

```
INDEX      { ieee8021PbbISidToVipISid }
 ::= { ieee8021PbbISidToVipTable 1 }

Ieee8021PbbISidToVipEntry ::= SEQUENCE {
    ieee8021PbbISidToVipISid
        IEEE8021PbbServiceIdentifier,
    ieee8021PbbISidToVipComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021PbbISidToVipPort
        IEEE8021BridgePortNumber
}

ieee8021PbbISidToVipISid OBJECT-TYPE
    SYNTAX      IEEE8021PbbServiceIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The I-SID of a VIP."
    ::= { ieee8021PbbISidToVipEntry 1 }

ieee8021PbbISidToVipComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The component identifier of the VIP to which this I-SID
        is associated."
    ::= { ieee8021PbbISidToVipEntry 2 }

ieee8021PbbISidToVipPort OBJECT-TYPE
    SYNTAX      IEEE8021BridgePortNumber
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The port number of the VIP to which this I-SID is associated."
    ::= { ieee8021PbbISidToVipEntry 3 }

-- =====
-- 12.16.4.1/2 Provider Instance Port (PIP) configuration
-- table
-- =====

ieee8021PbbPipTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbbPipEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains the parameters for each PIP, and
        can be used to configure the PIP port names. Entries
        in this table must be persistent over power up
        restart/reboot."
    REFERENCE   "12.16.4.1/2"
    ::= { ieee8021PbbProviderBackboneBridge 4 }

ieee8021PbbPipEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbbPipEntry
    MAX-ACCESS  not-accessible
    STATUS      current
```

```

DESCRIPTION
    "The parameters for a PIP.      "
INDEX      { ieee8021PbbPipIfIndex }
 ::= { ieee8021PbbPipTable 1 }

Ieee8021PbbPipEntry ::=-
SEQUENCE {
    ieee8021PbbPipIfIndex
        InterfaceIndex,
    ieee8021PbbPipBMACAddress
        MacAddress,
    ieee8021PbbPipName
        SnmpAdminString,
    ieee8021PbbPipIComponentId
        IEEE8021PbbComponentIdentifier,
    ieee8021PbbPipVipMap
        OCTET STRING,
    ieee8021PbbPipVipMap1
        OCTET STRING,
    ieee8021PbbPipVipMap2
        OCTET STRING,
    ieee8021PbbPipVipMap3
        OCTET STRING,
    ieee8021PbbPipVipMap4
        OCTET STRING,
    ieee8021PbbPipRowStatus
        RowStatus
}
ieee8021PbbPipIfIndex OBJECT-TYPE
SYNTAX      InterfaceIndex
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The number identifying the PIP."
 ::= { ieee8021PbbPipEntry 1 }

ieee8021PbbPipBMACAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The B-MAC used by this PIP for the B-SA."
 ::= { ieee8021PbbPipEntry 2 }

ieee8021PbbPipName OBJECT-TYPE
SYNTAX      SnmpAdminString (SIZE(0..32))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "A text string of local significance which identifies the
     PIP within a BEB."
DEFVAL { ''H }
 ::= { ieee8021PbbPipEntry 3 }

ieee8021PbbPipIComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  read-only
STATUS      current

```

```
DESCRIPTION
    "Identifies the I-component associated with this PIP."
 ::= { ieee8021PbbPipEntry 4 }

ieee8021PbbPipVipMap OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE(0..512))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object contains a bitmap indicating all the VIPs
     in the range 1 through 4094 that are associated with this
     PIP. The bits correspond to bridge port numbers in the
     range 1 through 4094. The high-order bit of the first
     octet corresponds to port number 1, and subsequent bits
     of the octet string correspond to subsequent port numbers.
     The following formula can be used to find the bit
     corresponding to a particular port number B:
        octet[(B-1)/8] & (1 >> ((B-1)%8))
     If the bit for a particular port number is 1, that VIP is
     associated with this PIP.

     The value of this object may be truncated to remove
     trailing octets of all zeroes."
DEFVAL { ''H }
 ::= { ieee8021PbbPipEntry 5 }

ieee8021PbbPipVipMap1 OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE(0..2048))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object contains a bitmap indicating all the VIPs
     in the range 4095 through 20478 that are associated with
     this PIP. The bits correspond to bridge port numbers in
     the range 4095 through 20478. The high-order bit of the first
     octet corresponds to port number 1, and subsequent bits
     of the octet string correspond to subsequent port numbers.
     The following formula can be used to find the bit
     corresponding to a particular port number B:
        octet[(B-4095)/8] & (1 >> ((B-4095)%8))
     If the bit for a particular port number is 1, that VIP is
     associated with this PIP.

     Note that ports numbers greater than 4094 cannot be used
     with xSTP.

     The value of this object may be truncated to remove
     trailing octets of all zeroes."
DEFVAL { ''H }
 ::= { ieee8021PbbPipEntry 6 }

ieee8021PbbPipVipMap2 OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE(0..2048))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object contains a bitmap indicating all the VIPs
     in the range 20479 through 36861 that are associated with
     this PIP. The bits correspond to bridge port numbers in
```

the range 20479 through 36861. The high-order bit of the first octet corresponds to port number 1, and subsequent bits of the octet string correspond to subsequent port numbers. The following formula can be used to find the bit corresponding to a particular port number B:

octet[(B-20479)/8] & (1 >> ((B-20479)%8))

If the bit for a particular port number is 1, that VIP is associated with this PIP.

Note that ports numbers greater than 4094 cannot be used with xSTP.

The value of this object may be truncated to remove trailing octets of all zeroes."

```
DEFVAL { 'H' }
 ::= { ieee8021PbbPipEntry 7 }
```

ieee8021PbbPipVipMap3 OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(0..2048))
 MAX-ACCESS read-create
 STATUS current
 DESCRIPTION

"This object contains a bitmap indicating all the VIPs in the range 36862 through 53245 that are associated with this PIP. The bits correspond to bridge port numbers in the range 36862 through 53245. The high-order bit of the first octet corresponds to port number 1, and subsequent bits of the octet string correspond to subsequent port numbers. The following formula can be used to find the bit corresponding to a particular port number B:

octet[(B-36862)/8] & (1 >> ((B-36862)%8))

If the bit for a particular port number is 1, that VIP is associated with this PIP.

Note that ports numbers greater than 4094 cannot be used with xSTP.

The value of this object may be truncated to remove trailing octets of all zeroes."

```
DEFVAL { 'H' }
 ::= { ieee8021PbbPipEntry 8 }
```

ieee8021PbbPipVipMap4 OBJECT-TYPE

SYNTAX OCTET STRING (SIZE(0..1537))
 MAX-ACCESS read-create
 STATUS current
 DESCRIPTION

"This object contains a bitmap indicating all the VIPs in the range 53246 through 65535 that are associated with this PIP. The bits correspond to bridge port numbers in the range 53246 through 65535. The high-order bit of the first octet corresponds to port number 1, and subsequent bits of the octet string correspond to subsequent port numbers. The following formula can be used to find the bit corresponding to a particular port number B:

octet[(B-53246)/8] & (1 >> ((B-53246)%8))

If the bit for a particular port number is 1, that VIP is associated with this PIP.

Note that ports numbers greater than 4094 cannot be used with xSTP.

The value of this object may be truncated to remove trailing octets of all zeroes."

```
DEFVAL { 'H }
 ::= { ieee8021PbbPipEntry 9 }
```

```
ieee8021PbbPipRowStatus OBJECT-TYPE
 SYNTAX      RowStatus
 MAX-ACCESS  read-create
 STATUS      current
 DESCRIPTION
 "Indicates the status of an entry in this table, and is used
 to create/delete entries.
```

The object ieee8021PbbPipBMACAddress must be set before this object can be made active(1).

The value of ieee8021PbbPipBMACAddress cannot be modified while this object is active(1)."

```
 ::= { ieee8021PbbPipEntry 10 }
```

```
-- =====
-- 12.16.4.1      Provider Instance Port (PIP)
--          Priority Table
-- =====
```

```
ieee8021PbbPipPriorityTable OBJECT-TYPE
 SYNTAX      SEQUENCE OF Ieee8021PbbPipPriorityEntry
 MAX-ACCESS  not-accessible
 STATUS      current
 DESCRIPTION
 "A table that contains information about every PIP that
 is associated with this PBB."
 REFERENCE   "12.16.4.1"
 ::= { ieee8021PbbProviderBackboneBridge 5 }
```

```
ieee8021PbbPipPriorityEntry OBJECT-TYPE
 SYNTAX      Ieee8021PbbPipPriorityEntry
 MAX-ACCESS  not-accessible
 STATUS      current
 DESCRIPTION
 "A list of Default User Priorities for each PIP of a
 PBB. This is indexed by ieee8021PbbPipIfIndex."
 AUGMENTS { ieee8021PbbPipEntry }
 ::= { ieee8021PbbPipPriorityTable 1 }
```

```
Ieee8021PbbPipPriorityEntry ::=
 SEQUENCE {
     ieee8021PbbPipPriorityCodePointSelection
         IEEE8021PriorityCodePoint,
     ieee8021PbbPipUseDEI
         TruthValue,
     ieee8021PbbPipRequireDropEncoding
         TruthValue
 }
```

```
ieee8021PbbPipPriorityCodePointSelection OBJECT-TYPE
```

```

SYNTAX      IEEE8021PriorityCodePoint
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
  " This object identifies the rows in the PCP encoding and
  decoding tables that are used to remark frames on this
  PIP if this remarking is enabled."
REFERENCE   "12.16.4.5, 12.16.4.6"
 ::= { ieee8021PbbPipPriorityEntry 1 }

ieee8021PbbPipUseDEI OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
  "If the Use_DEI is set to true(1) for the PIP then the
  drop_eligible parameter is encoded in the DEI of transmitted
  frames, and the drop_eligible parameter shall be true(1) for a
  received frame if the DEI is set in the VLAN tag or the Priority
  Code Point Decoding Table indicates drop_eligible True for
  the received PCP value. If the Use_DEI parameter is false(2),
  the DEI shall be transmitted as zero and ignored on receipt.
  The default value of the Use_DEI parameter is false(2)."
REFERENCE   "12.16.4.11, 12.16.412"
 ::= { ieee8021PbbPipPriorityEntry 2 }

ieee8021PbbPipRequireDropEncoding OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
  "If a Bridge supports encoding or decoding of drop_eligible
  from the PCP field of a VLAN tag (6.7.3) on any of its PIPs,
  then it shall implement a Boolean parameter Require Drop
  Encoding on each of its PIPs with default value false(2). If
  Require Drop Encoding is True and the PIP cannot
  encode particular priorities with drop_eligible, then frames
  queued with those priorities and drop_eligible true(1) shall
  be discarded and not transmitted."
REFERENCE   "12.16.4.13, 12.16.4.14"
DEFVAL { false }
 ::= { ieee8021PbbPipPriorityEntry 3 }

-- =====
-- 12.16.4.7/8 Provider Instance Port (PIP)
--          Priority Code Point (PCP) Decoding Table
-- =====

ieee8021PbbPipDecodingTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbbPipDecodingEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
  "A table that contains information about Priority Code
  Point Decoding Table for a PIP of a provider bridge.
  Alternative values for each table are specified as rows
  in Table 6-4 (6.7.3), with each alternative labeled by
  the number of distinct priorities that can be communicated,
  and the number of these for which drop precedence can

```

```
    be communicated. All writable objects in this table must
    be persistent over power up restart/reboot."
REFERENCE    "12.16.4.7, 12.16.4.8"
 ::= { ieee8021PbbProviderBackboneBridge 6 }

ieee8021PbbPipDecodingEntry OBJECT-TYPE
SYNTAX      Ieee8021PbbPipDecodingEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A list of objects containing Priority Code Point Decoding
     information for a PIP of a provider bridge."
INDEX { ieee8021PbbPipIfIndex,
        ieee8021PbbPipDecodingPriorityCodePointRow,
        ieee8021PbbPipDecodingPriorityCodePoint }
 ::= { ieee8021PbbPipDecodingTable 1 }

Ieee8021PbbPipDecodingEntry ::= SEQUENCE {
    ieee8021PbbPipDecodingPriorityCodePointRow
        IEEE8021PriorityCodePoint,
    ieee8021PbbPipDecodingPriorityCodePoint
        Integer32,
    ieee8021PbbPipDecodingPriority
        IEEE8021PriorityValue,
    ieee8021PbbPipDecodingDropEligible
        TruthValue
}

ieee8021PbbPipDecodingPriorityCodePointRow OBJECT-TYPE
SYNTAX      IEEE8021PriorityCodePoint
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The specific row in Table 6-3 (6.7.3) indicating the PCP."
 ::= { ieee8021PbbPipDecodingEntry 1 }

ieee8021PbbPipDecodingPriorityCodePoint OBJECT-TYPE
SYNTAX      Integer32 (0..7)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The specific PCP value in Table 6-3 (6.7.3)."
 ::= { ieee8021PbbPipDecodingEntry 2 }

ieee8021PbbPipDecodingPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The specific priority value in Table 6-3 (6.7.3)."
REFERENCE    "12.6.2.8, 12.6.2.9"
 ::= { ieee8021PbbPipDecodingEntry 3 }

ieee8021PbbPipDecodingDropEligible OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The drop eligibility value in Table 6-3 (6.7.3)."
```

```

REFERENCE    "12.6.2.8, 12.6.2.9"
 ::= { ieee8021PbbPipDecodingEntry 4 }

-- =====
-- 12.16.4.9/10 Provider Instance Port (PIP)
--      Priority Code Point (PCP) Encoding Table
-- =====

ieee8021PbbPipEncodingTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbbPipEncodingEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table that contains information about Priority Code
Point Decoding Table for a PIP of a provider bridge.
Alternative values for each table are specified as rows
in Table 6-3 (6.7.3), with each alternative labeled by
the number of distinct priorities that can be communicated,
and the number of these for which drop precedence can be
communicated. All writable objects in this table must be
persistent over power up restart/reboot."
REFERENCE    "12.16.4.9, 12.16.4.10"
 ::= { ieee8021PbbProviderBackboneBridge 7 }

ieee8021PbbPipEncodingEntry OBJECT-TYPE
SYNTAX      Ieee8021PbbPipEncodingEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A list of objects containing Priority Code Point Encoding
information for a PIP of a provider bridge."
INDEX { ieee8021PbbPipIfIndex,
        ieee8021PbbPipEncodingPriorityCodePointRow,
        ieee8021PbbPipEncodingPriorityCodePoint,
        ieee8021PbbPipEncodingDropEligible }
 ::= { ieee8021PbbPipEncodingTable 1 }

Ieee8021PbbPipEncodingEntry ::= SEQUENCE {
    ieee8021PbbPipEncodingPriorityCodePointRow
        IEEE8021PriorityCodePoint,
    ieee8021PbbPipEncodingPriorityCodePoint
        Integer32,
    ieee8021PbbPipEncodingDropEligible
        TruthValue,
    ieee8021PbbPipEncodingPriority
        IEEE8021PriorityValue
}

ieee8021PbbPipEncodingPriorityCodePointRow OBJECT-TYPE
SYNTAX      IEEE8021PriorityCodePoint
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"The specific row in Table 6-3 (6.7.3) indicating the PCP row.
(i.e. 8P0D, 7P1D, 6P2D, 5P3D)"
 ::= { ieee8021PbbPipEncodingEntry 1 }

ieee8021PbbPipEncodingPriorityCodePoint OBJECT-TYPE
SYNTAX      Integer32 (0..7)

```

```
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The specific row in Table 6-3 (6.7.3) indicating the PCP.
     (i.e., 0,1,2,3,4,5,6,7)."
 ::= { ieee8021PbbPipEncodingEntry 2 }

ieee8021PbbPipEncodingDropEligible OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "The specific row in Table 6-3 (6.7.3) indicating the drop
     eligibility. A value of true(1) means eligible for drop."
 ::= { ieee8021PbbPipEncodingEntry 3 }

ieee8021PbbPipEncodingPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS read-write
STATUS      current
DESCRIPTION
    "The encoding priority in Table 6-3 (6.7.3)."
REFERENCE   "12.6.2.10, 12.6.2.11"
 ::= { ieee8021PbbPipEncodingEntry 4 }

-- =====
-- 12.16.4.3/4 Virtual Instance Port (VIP) to Provider
--           Instance Port (PIP) mapping table
-- =====

ieee8021PbbVipToPipMappingTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbbVipToPipMappingEntry
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    "This table lists VIPs and the PIP to which each is
     associated, and allows the PIP associated with each
     VIP to be configured. Entries in this table must be
     persistent over power up restart/reboot."
REFERENCE   "12.16.4.3/4"
 ::= { ieee8021PbbProviderBackboneBridge 8 }

ieee8021PbbVipToPipMappingEntry OBJECT-TYPE
SYNTAX      Ieee8021PbbVipToPipMappingEntry
MAX-ACCESS not-accessible
STATUS      current
DESCRIPTION
    " The PIP is identified by the value of the
     ieee8021PbbVipToPipNumber. This value may be used to
     index the ieee8021PbbPipTable to set or retrieve the
     PIP's configuration information"
INDEX       { ieee8021BridgeBasePortComponentId,
              ieee8021BridgeBasePort }
 ::= { ieee8021PbbVipToPipMappingTable 1 }

Ieee8021PbbVipToPipMappingEntry ::=
SEQUENCE {
    ieee8021PbbVipToPipMappingPipIfIndex
    InterfaceIndex,
```

```

        ieee8021PbbVipToPipMappingStorageType
            StorageType,
        ieee8021PbbVipToPipMappingRowStatus
            RowStatus
    }

ieee8021PbbVipToPipMappingPipIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The PIP's interface number."
    ::= { ieee8021PbbVipToPipMappingEntry 1 }

ieee8021PbbVipToPipMappingStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Indicates the storage type of this entry. An entry whose
         storage type is permanent(4) need not allow write access to
         other columns in that entry."
    ::= { ieee8021PbbVipToPipMappingEntry 2 }

ieee8021PbbVipToPipMappingRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Indicates the status of an entry in this table, and is used
         to create/delete entries.

The corresponding instance of ieee8021PbbVipToPipMappingPipIfIndex
must be set before this object can be made active(1).

The corresponding instance of ieee8021PbbVipToPipMappingPipIfIndex
may not be changed while this object is active(1)."
    ::= { ieee8021PbbVipToPipMappingEntry 3 }

-- =====
-- 12.16.5.1/2 Service Mapping configuration table
-- =====

ieee8021PbbCBPServiceMappingTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbbCBPServiceMappingEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The CBP table of I-SID values (6.11). The contents of this
         table are not persistent over power up restart/reboot."
    REFERENCE   "12.16.5.1/2"
    ::= { ieee8021PbbProviderBackboneBridge 9 }

ieee8021PbbCBPServiceMappingEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbbCBPServiceMappingEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Each entry includes the B-VID to carry and optionally an

```

I-SID for mapping I-SIDs normally used at a Peer E-NNI (6.11, 26.6.2). The table is indexed by the component ID of the relevant B-Component of the PBB, the Bridge port number of the CBP on that Bcomponent, and the I-SID for the service. "

INDEX { ieee8021BridgeBasePortComponentId,
 ieee8021BridgeBasePort,
 ieee8021PbbCBPServiceMappingBackboneSid }
 ::= { ieee8021PbbCBPServiceMappingTable 1 }

Ieee8021PbbCBPServiceMappingEntry ::=
SEQUENCE {
 ieee8021PbbCBPServiceMappingBackboneSid
 IEEE8021PbbServiceIdentifier,
 ieee8021PbbCBPServiceMappingBVid
 VlanId,
 ieee8021PbbCBPServiceMappingDefaultBackboneDest
 MacAddress,
 ieee8021PbbCBPServiceMappingType
 IEEE8021PbbIngressEgress,
 ieee8021PbbCBPServiceMappingLocalSid
 IEEE8021PbbServiceIdentifierOrUnassigned,
 ieee8021PbbCBPServiceMappingRowStatus
 RowStatus
}

ieee8021PbbCBPServiceMappingBackboneSid OBJECT-TYPE
SYNTAX IEEE8021PbbServiceIdentifier
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "The I-SID which will be transmitted over the PBBN."
 ::= { ieee8021PbbCBPServiceMappingEntry 1 }

ieee8021PbbCBPServiceMappingBVid OBJECT-TYPE
SYNTAX VlanId
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "The B-VID which will carry this service instance."
 ::= { ieee8021PbbCBPServiceMappingEntry 2 }

ieee8021PbbCBPServiceMappingDefaultBackboneDest OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "A default destination B-MAC for the CBP to use."
 ::= { ieee8021PbbCBPServiceMappingEntry 3 }

ieee8021PbbCBPServiceMappingType OBJECT-TYPE
SYNTAX IEEE8021PbbIngressEgress
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "Used for Pt-MPt service where ingress or egress limiting
 is desired."
 ::= { ieee8021PbbCBPServiceMappingEntry 4 }

```

ieee8021PbbCBPServiceMappingLocalSid OBJECT-TYPE
    SYNTAX      IEEE8021PbbServiceIdentifierOrUnassigned
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The I-SID value used in frames transmitted and received through
         this CustomerBackbonePort."
    DEFVAL { 1 }
    ::= { ieee8021PbbCBPServiceMappingEntry 5 }

ieee8021PbbCBPServiceMappingRowStatus OBJECT-TYPE
    SYNTAX      RowStatus
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "Indicates the status of an entry in this table, and is used
         to create/delete entries.

The corresponding instances of the following objects
must be set before this object can be made active(1):
    ieee8021PbbCBPServiceMappingBVid
    ieee8021PbbCBPServiceMappingDefaultBackboneDest
    ieee8021PbbCBPServiceMappingType

The corresponding instances of the following objects
may not be changed while this object is active(1):
    ieee8021PbbCBPServiceMappingBVid
    ieee8021PbbCBPServiceMappingDefaultBackboneDest
    ieee8021PbbCBPServiceMappingType
    ieee8021PbbCBPServiceMappingLocalSid"
    ::= { ieee8021PbbCBPServiceMappingEntry 6 }

-- =====
-- CBP port creation/deletion table
-- =====

ieee8021PbbCbpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021PbbCbpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table is used to dynamically create/delete Customer
         Backbone Ports in a PBB.

Entries in this table must be persistent across reinitializations
of the management system."
    REFERENCE   "17.5.3.4"
    ::= { ieee8021PbbProviderBackboneBridge 10 }

ieee8021PbbCbpEntry OBJECT-TYPE
    SYNTAX      Ieee8021PbbCbpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry representing a dynamically created CBP in a PBB."
    INDEX       { ieee8021BridgeBasePortComponentId,
                  ieee8021BridgeBasePort }
    ::= { ieee8021PbbCbpTable 1 }

```

```
Ieee8021PbbCbpEntry ::=  
SEQUENCE {  
    ieee8021PbbCbpRowStatus  
    RowStatus  
}  
  
ieee8021PbbCbpRowStatus OBJECT-TYPE  
SYNTAX      RowStatus  
MAX-ACCESS  read-create  
STATUS      current  
DESCRIPTION  
    "This object is used for creation/deletion of entries in  
    this table."  
::= { ieee8021PbbCbpEntry 1 }  
  
-- =====  
-- Conformance Information  
-- =====  
  
ieee8021PbbGroups  
OBJECT IDENTIFIER ::= { ieee8021PbbConformance 1 }  
ieee8021PbbCompliances  
OBJECT IDENTIFIER ::= { ieee8021PbbConformance 2 }  
  
-- =====  
-- Units of conformance  
-- =====  
  
ieee8021PbbBackboneEdgeBridgeGroup OBJECT-GROUP  
OBJECTS {  
    ieee8021PbbBackboneEdgeBridgeAddress,  
    ieee8021PbbBackboneEdgeBridgeName,  
    ieee8021PbbNumberOfIComponents,  
    ieee8021PbbNumberOfBComponents,  
    ieee8021PbbNumberOfBebPorts  
}  
STATUS      current  
DESCRIPTION  
    "The collection of objects used to represent a Backbone  
    Edge Bridge."  
::= { ieee8021PbbGroups 1 }  
  
ieee8021PbbVipGroup OBJECT-GROUP  
OBJECTS {  
    ieee8021PbbVipPipIfIndex,  
    ieee8021PbbVipISid,  
    ieee8021PbbVipDefaultDstMAC,  
    ieee8021PbbVipType,  
    ieee8021PbbVipRowStatus,  
    ieee8021PbbISidToVipComponentId,  
    ieee8021PbbISidToVipPort  
}  
STATUS      current  
DESCRIPTION  
    "The collection of objects used to represent a Virtual  
    Instance Port (VIP)."  
::= { ieee8021PbbGroups 2 }  
  
ieee8021PbbPipGroup OBJECT-GROUP
```

```

OBJECTS {
    ieee8021PbbNextAvailablePipIfIndex,
    ieee8021PbbPipBMACAddress,
    ieee8021PbbPipName,
    ieee8021PbbPipComponentId,
    ieee8021PbbPipVipMap,
    ieee8021PbbPipVipMap1,
    ieee8021PbbPipVipMap2,
    ieee8021PbbPipVipMap3,
    ieee8021PbbPipVipMap4,
    ieee8021PbbPipRowStatus
}
STATUS      current
DESCRIPTION
    "The collection of objects used to represent a Provider
     Instance Port (PIP)."
::= { ieee8021PbbGroups 3 }

ieee8021PbbDefaultPriorityGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbPipPriorityCodePointSelection,
    ieee8021PbbPipUseDEI,
    ieee8021PbbPipRequireDropEncoding
}
STATUS      current
DESCRIPTION
    "A collection of objects defining the User Priority
     applicable to each port for media that do not support
     native User Priority."
::= { ieee8021PbbGroups 4 }

ieee8021PbbPipDecodingGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbPipDecodingPriority,
    ieee8021PbbPipDecodingDropEligible
}
STATUS      current
DESCRIPTION
    "A collection of objects providing statistics counters for
     decoding priority and drop eligibility for bridge ports."
::= { ieee8021PbbGroups 5 }

ieee8021PbbPipEncodingGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbPipEncodingPriority
}
STATUS      current
DESCRIPTION
    "A collection of objects providing statistics counters for
     encoding priority and drop eligibility for bridge ports."
::= { ieee8021PbbGroups 6 }

ieee8021PbbVipToPipMappingGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbVipToPipMappingPipIfIndex,
    ieee8021PbbVipToPipMappingStorageType,
    ieee8021PbbVipToPipMappingRowStatus
}
STATUS      current

```

```
DESCRIPTION
    "The collection of objects used to represent the mapping
     of a VIP to a PIP."
 ::= { ieee8021PbbGroups 7 }

ieee8021PbbCBPServiceMappingGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbCBPServiceMappingBVid,
    ieee8021PbbCBPServiceMappingDefaultBackboneDest,
    ieee8021PbbCBPServiceMappingType,
    ieee8021PbbCBPServiceMappingLocalSid,
    ieee8021PbbCBPServiceMappingRowStatus
}
STATUS      current
DESCRIPTION
    "The collection of objects used to represent a service instance."
 ::= { ieee8021PbbGroups 8 }

ieee8021PbbDynamicCbpGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbCbpRowStatus
}
STATUS      current
DESCRIPTION
    "The collection of objects used to dynamically create/delete
     CBPs in a PBB."
 ::= { ieee8021PbbGroups 9 }

ieee8021PbbVipPbbTeGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbVipEnableConnectionId
}
STATUS      current
DESCRIPTION
    "The collection of objects specific to PBB bridges operating
     in a PBB-TE aware manner."
 ::= { ieee8021PbbGroups 10 }

-- =====
-- Compliance statements
-- =====

ieee8021PbbCompliance MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for devices supporting Provider
     Backbone Bridging as defined in IEEE 802.1ah."
MODULE
MANDATORY-GROUPS {
    ieee8021PbbBackboneEdgeBridgeGroup,
    ieee8021PbbVipGroup,
    ieee8021PbbPipGroup,
    ieee8021PbbVipToPipMappingGroup,
    ieee8021PbbCBPServiceMappingGroup,
    ieee8021PbbDynamicCbpGroup
}

GROUP      ieee8021PbbDefaultPriorityGroup
DESCRIPTION
```

"This group is mandatory only for devices supporting the priority forwarding operations defined by the extended bridge services with media types, such as Ethernet, that do not support native User Priority."

GROUP ieee8021PbbPipDecodingGroup
DESCRIPTION
 "This group is optional and supports Priority Code Point Decoding Table for a PIP of a provider bridge."

GROUP ieee8021PbbPipEncodingGroup
DESCRIPTION
 "This group is optional and supports Priority Code Point Encoding Table for a PIP of a provider bridge."

OBJECT ieee8021PbbPipName
MIN-ACCESS read-only
DESCRIPTION
 "Read-write access for this object is not required."

OBJECT ieee8021PbbPipVipMap
MIN-ACCESS read-only
DESCRIPTION
 "Read-write access for this object is not required."

OBJECT ieee8021PbbVipToPipMappingPipIfIndex
MIN-ACCESS read-only
DESCRIPTION
 "Read-write access for this object is not required."

OBJECT ieee8021PbbCBPServiceMappingBVid
MIN-ACCESS not-accessible
DESCRIPTION
 "Read-only and read-write access for this object are optional."

OBJECT ieee8021PbbCBPServiceMappingDefaultBackboneDest
MIN-ACCESS not-accessible
DESCRIPTION
 "Read-only and read-write access for this object are optional."

OBJECT ieee8021PbbCBPServiceMappingLocalSid
MIN-ACCESS not-accessible
DESCRIPTION
 "Read-only and read-write access for this object are optional."

::= { ieee8021PbbCompliances 1 }

ieee8021PbbWithPbbTeCompliance MODULE-COMPLIANCE
 STATUS current
 DESCRIPTION
 "The compliance statement for devices supporting Provider Backbone Bridging with traffic engineering as defined in IEEE 802.1ah and IEEE 802.1Qay."
 MODULE
 MANDATORY-GROUPS {
 ieee8021PbbBackboneEdgeBridgeGroup,
 ieee8021PbbVipGroup,
 ieee8021PbbPipGroup,
 ieee8021PbbVipToPipMappingGroup,
 }

```
        ieee8021PbbCBPServiceMappingGroup,
        ieee8021PbbDynamicCbpGroup,
ieee8021PbbVipPbbTeGroup
}

GROUP      ieee8021PbbDefaultPriorityGroup
DESCRIPTION
"This group is mandatory only for devices supporting
the priority forwarding operations defined by the
extended bridge services with media types, such as
Ethernet, that do not support native User Priority."

GROUP      ieee8021PbbPipDecodingGroup
DESCRIPTION
"This group is optional and supports Priority Code Point
Decoding Table for a PIP of a provider bridge."

GROUP      ieee8021PbbPipEncodingGroup
DESCRIPTION
"This group is optional and supports Priority Code Point
Encoding Table for a PIP of a provider bridge."

OBJECT ieee8021PbbPipName
MIN-ACCESS read-only
DESCRIPTION
"Read-write access for this object is not required."

OBJECT ieee8021PbbPipVipMap
MIN-ACCESS read-only
DESCRIPTION
"Read-write access for this object is not required."

OBJECT ieee8021PbbVipToPipMappingPipIfIndex
MIN-ACCESS read-only
DESCRIPTION
"Read-write access for this object is not required."

OBJECT ieee8021PbbCBPServiceMappingBVid
MIN-ACCESS not-accessible
DESCRIPTION
"Read-only and read-write access for this object are optional."

OBJECT ieee8021PbbCBPServiceMappingDefaultBackboneDest
MIN-ACCESS not-accessible
DESCRIPTION
"Read-only and read-write access for this object are optional."

OBJECT ieee8021PbbCBPServiceMappingLocalSid
MIN-ACCESS not-accessible
DESCRIPTION
"Read-only and read-write access for this object are optional."


::= { ieee8021PbbCompliances 2 }

END
```

17.7.9 Definitions for the IEEE8021-DDCFM MIB module

```

IEEE8021-DDCFM-MIB DEFINITIONS ::= BEGIN

-- *****
-- IEEE802.1Qaw DDCFM MIB
-- *****

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Unsigned32 FROM SNMPv2-SMI -- [RFC2578]
    TruthValue, RowStatus,
    MacAddress FROM SNMPv2-TC -- [RFC2579]
    MODULE-COMPLIANCE, OBJECT-GROUP FROM SNMPv2-CONF -- [RFC2580]
    InterfaceIndex FROM IF-MIB -- [RFC2863]
    VlanIdOrNone FROM Q-BRIDGE-MIB -- [RFC4363]
    Dot1agCfmMDLevel, Dot1agCfmMpDirection FROM IEEE8021-CFM-MIB
    ieee802dot1mibs FROM IEEE8021-TC-MIB
;

ieee8021DdcfmMIB MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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: Linda Dunbar
     Postal: C/O IEEE 802.1 Working Group
              IEEE Standards Association
              445 Hoes Lane
              P.O. Box 1331
              Piscataway
              NJ 08855-1331
              USA
     E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
    "module for managing Data Dependent and Data Driven Connectivity Fault
Management.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
REVISION      "201102270000Z" -- February 27, 2011
DESCRIPTION
    "Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."
REVISION "200904060000Z" -- 04/06/2009 00:00GMT
DESCRIPTION
    "Included in IEEE802.1Qaw D5.0
     Copyright (c) IEEE"
::= {ieee802dot1mibs 11}

```

```
ieee8021MIBObjects OBJECT IDENTIFIER ::= { ieee8021DdcfMIB 1 }
ieee8021DdcfMConformance OBJECT IDENTIFIER ::= { ieee8021DdcfMIB 2 }

-- ****
-- Groups in the DDCFM MIB Module
-- ****

ieee8021DdcfMStack OBJECT IDENTIFIER ::= {ieee8021MIBObjects 1}
ieee8021DdcfMRR OBJECT IDENTIFIER ::= {ieee8021MIBObjects 2}
ieee8021DdcfMRFMReceiver OBJECT IDENTIFIER ::= {ieee8021MIBObjects 3}
ieee8021DdcfMDr OBJECT IDENTIFIER ::= {ieee8021MIBObjects 4}
ieee8021DdcfSFMOOriginator OBJECT IDENTIFIER ::= {ieee8021MIBObjects 5}

-- ****
-- The DDCFM Stack Table
-- ****

ieee8021DdcfMStackTable OBJECT-TYPE
    SYNTAX SEQUENCE OF Ieee8021DdcfMStackEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The DDCFM Stack object table. This table is for operator to
         retrieve all the DDCFM entities defined on a specified interface.
         This table is created by default."
    REFERENCE
        "clauses 12.18.1"
    ::= { ieee8021DdcfMStack 1 }

ieee8021DdcfMStackEntry OBJECT-TYPE
    SYNTAX Ieee8021DdcfMStackEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The DDCFM Stack Table. "
    INDEX { ieee8021DdcfMStackIfIndex }
    ::= { ieee8021DdcfMStackTable 1 }

Ieee8021DdcfMStackEntry ::= SEQUENCE {
    ieee8021DdcfMStackIfIndex InterfaceIndex,
    ieee8021DdcfMStackRrMdLevel Dot1agCfmMDLevel,
    ieee8021DdcfMStackRrDirection Dot1agCfmMpDirection,
    ieee8021DdcfMStackRFMReceiverMdLevel Dot1agCfmMDLevel,
    ieee8021DdcfMStackDrMdLevel Dot1agCfmMDLevel,
    ieee8021DdcfMStackDrVlanIdOrNone VlanIdOrNone,
    ieee8021DdcfMStackSFMOOriginatorMdLevel Dot1agCfmMDLevel,
    ieee8021DdcfMStackSFMOOriginatorVlanIdOrNone VlanIdOrNone,
    ieee8021DdcfMStackSFMOOriginatorDirection Dot1agCfmMpDirection
}

ieee8021DdcfMStackIfIndex OBJECT-TYPE
    SYNTAX InterfaceIndex
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "This object is the interface index of the interface either a
         bridge port, or an aggregated IEEE802.3 port within a Bridge
         Port. When the ifIndex value corresponds to the ifIndex of a
         Bridge Port, the value in this column must match the value in the
```

```
ieee8021BridgeBasePortIfIndex column for the bridge port."  
REFERENCE  
    "clause 12.18.1.1.2 a.1"  
::= { ieee8021DdcfStackEntry 1 }

ieee8021DdcfStackRrMdLevel OBJECT-TYPE  
SYNTAX Dot1agCfmMDLevel  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "MD level of the Reflection Responder managed object."  
REFERENCE  
    "clause 12.18.1.1.3 b.1"  
::= { ieee8021DdcfStackEntry 2 }

ieee8021DdcfStackRrDirection OBJECT-TYPE  
SYNTAX Dot1agCfmMpDirection  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "The direction in which the RR faces."  
REFERENCE  
    "clause 12.18.1.1.3 b.1"  
::= { ieee8021DdcfStackEntry 3 }

ieee8021DdcfStackRFMReceiverMdLevel OBJECT-TYPE  
SYNTAX Dot1agCfmMDLevel  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "MD level of the RFM Receiver MO configured on the interface."  
REFERENCE  
    "clause 12.18.1.1.3 b.2"  
::= { ieee8021DdcfStackEntry 4 }

ieee8021DdcfStackDrMdLevel OBJECT-TYPE  
SYNTAX Dot1agCfmMDLevel  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "MD level of the Deflection Responder managed object."  
REFERENCE  
    "clause 12.18.1.1.3 b.3"  
::= { ieee8021DdcfStackEntry 5 }

ieee8021DdcfStackDrVlanIdOrNone OBJECT-TYPE  
SYNTAX VlanIdOrNone  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
    "The MA of the DR configured on the interface."  
REFERENCE  
    "clause 12.18.1.1.3 b.3"  
::= { ieee8021DdcfStackEntry 6 }

ieee8021DdcfStackSFMOriginatorMdLevel OBJECT-TYPE  
SYNTAX Dot1agCfmMDLevel  
MAX-ACCESS read-only  
STATUS current
```

```
DESCRIPTION
    "MD level of the SFM Originator MO configured on the interface."
REFERENCE
    "clause 12.18.1.1.3 b.4"
 ::= { ieee8021DdcfStackEntry 7 }

ieee8021DdcfStackSFMOiginatorVlanIdOrNone OBJECT-TYPE
SYNTAX VlanIdOrNone
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "The MA of the SFM Originator configured on the interface."
REFERENCE
    "clause 12.18.1.1.3 b.4"
 ::= { ieee8021DdcfStackEntry 8 }

ieee8021DdcfStackSFMOiginatorDirection OBJECT-TYPE
SYNTAX Dot1agCfmMpDirection
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "The direction of which the SFM Originator is facing."
REFERENCE
    "clause 12.18.1.1.3 b.4"
 ::= { ieee8021DdcfStackEntry 9 }

-- ****
-- The Reflection Responder Table
-- ****
ieee8021DdcfRrTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021DdcfRrEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The Reflection Responder MIB object table. Each row in the table
     represents a different Reflection Responder. All rows in this
     table persist across a system restart, however after such a restart
     the value of the ActivationStatus column will be false."
REFERENCE
    "clauses 12.18.2"
 ::= { ieee8021DdcfRr 1 }

ieee8021DdcfRrEntry OBJECT-TYPE
SYNTAX Ieee8021DdcfRrEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The Reflection Responder. Each entry associated with a Reflection
     Responder."
INDEX { ieee8021DdcfRrIfIndex,
        ieee8021DdcfRrMdIndex,
        ieee8021DdcfRrDirection
       }
 ::= { ieee8021DdcfRrTable 1 }

Ieee8021DdcfRrEntry ::= SEQUENCE {
    ieee8021DdcfRrIfIndex InterfaceIndex,
    ieee8021DdcfRrMdIndex Dot1agCfmMDLevel,
    ieee8021DdcfRrDirection Dot1agCfmMpDirection,
```

```

ieee8021DdcfmRrPrimaryVlanIdOrNone VlanIdOrNone,
ieee8021DdcfmRrFilter OCTET STRING,
ieee8021DdcfmRrSamplingInterval Unsigned32,
ieee8021DdcfmRrTargetAddress MacAddress,
ieee8021DdcfmRrContinueFlag TruthValue,
ieee8021DdcfmRrDuration Unsigned32,
ieee8021DdcfmRrDurationInTimeFlag TruthValue,
ieee8021DdcfmRrVlanPriority Integer32,
ieee8021DdcfmRrVlanDropEligible TruthValue,
ieee8021DdcfmRrFloodingFlag TruthValue,
ieee8021DdcfmRrTruncationFlag TruthValue,
ieee8021DdcfmRrActivationStatus TruthValue,
ieee8021DdcfmRrRemainDuration Unsigned32,
ieee8021DdcfmRrNextRfmTransID Unsigned32,
ieee8021DdcfmRrRowStatus RowStatus
}

ieee8021DdcfmRrIfIndex OBJECT-TYPE
SYNTAX InterfaceIndex
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"This object is the interface index of the interface either a
bridge port, or an aggregated IEEE802.3 port within a Bridge
Port, on which Reflection Responder is defined.
When the ifIndex value corresponds to the ifIndex of a
Bridge Port, the value in this column must match the value in the
ieee8021BridgeBasePortIfIndex column for the bridge port."
REFERENCE
"clause 12.18.2.1.2 a.1"
 ::= { ieee8021DdcfmRrEntry 1 }

ieee8021DdcfmRrMdIndex OBJECT-TYPE
SYNTAX Dot1agCfmMDLevel
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"MD level of the Reflection Responder managed object."
REFERENCE
"clause 12.18.2.1.2 a.2"
 ::= { ieee8021DdcfmRrEntry 2 }

ieee8021DdcfmRrDirection OBJECT-TYPE
SYNTAX Dot1agCfmMpDirection
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The direction in which the RR faces."
REFERENCE
"clause 12.18.2.1.2 a.3"
 ::= { ieee8021DdcfmRrEntry 3 }

ieee8021DdcfmRrPrimaryVlanIdOrNone OBJECT-TYPE
SYNTAX VlanIdOrNone
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The VID to be used on RFM frames."
REFERENCE

```

```
        "clause 12.18.2.2.2 b.1"
DEFVAL { 0 }
 ::= { ieee8021DdcfmrEntry 4 }

ieee8021DdcfmrFilter OBJECT-TYPE
SYNTAX OCTET STRING (SIZE(0..1500))
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"A pattern string specifies what data frames are selected to be
reflected. Below are the primary Reflection Filters all
Implementers should support. Multiple primary filters can be
combined together by
&& (and), || (or), or !(negation).
1) All;
2) VID= vid;
3) I-SID = x;
4) DA = xx.xx.xx.xx.xx.xx;
5) SA = xx.xx.xx.xx.xx.xx;
6) EtherType =xx.
For the reason that this management object allows a max size of
1500, messages carrying this object may be fragmented on some
segments."
REFERENCE
"clause 12.18.2.2.2 b.2"
 ::= { ieee8021DdcfmrEntry 5 }

ieee8021DdcfmrSamplingInterval OBJECT-TYPE
SYNTAX Unsigned32
UNITS "milliseconds"
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"Indicates a time interval in which only the first frame matching
the filter conditions is reflected."
REFERENCE
"clause 12.18.2.2.2 b.3"
 ::= { ieee8021DdcfmrEntry 6 }

ieee8021DdcfmrTargetAddress OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The Reflection Target Address, which is a MAC address to which the
reflected frames are targeted. Only individual address is allowed
for the Reflection Target Address.
If not specified, the source_address of the selected data frame is
used for Reflection Target Address."
REFERENCE
"clause 12.18.2.2.2 b.4"
 ::= { ieee8021DdcfmrEntry 7 }

ieee8021DdcfmrContinueFlag OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"True indicates that the selected data frames are to be continued
```

towards the DA specified in the frame header.
 False indicates that the selected data frames are terminated."

REFERENCE
 "clause 12.18.2.2.2 b.5"
DEFVAL { true }
::= { ieee8021DdcfmrEntry 8 }

ieee8021DdcfmrDuration OBJECT-TYPE
SYNTAX Unsigned32
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "Duration of time in the unit of seconds or the number of frames to be reflected, for Reflection Responder to remain active after activation; Minimum 2 octets (65536 seconds) are needed for the duration of time;"

REFERENCE
 "clause 12.18.2.2.2 b.7"
::= { ieee8021DdcfmrEntry 9 }

ieee8021DdcfmrDurationInTimeFlag OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "True indicates that duration is in seconds;
 False indicates that duration is by the total number of frames reflected."

REFERENCE
 "clause 12.18.2.2.2 b.6"
DEFVAL { true }
::= { ieee8021DdcfmrEntry 10 }

ieee8021DdcfmrVlanPriority OBJECT-TYPE
SYNTAX Integer32(0..7)
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "Priority, 3 bit value to be used in the VLAN tag, to be used in the transmitted encapsulated frames. The default value is the highest priority."

REFERENCE
 "clause 12.18.2.2.2 b.9"
DEFVAL { 7 }
::= { ieee8021DdcfmrEntry 11 }

ieee8021DdcfmrVlanDropEligible OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
 "It indicates that drop_eligible bit value to be used in the VLAN tag which to be used in the transmitted encapsulated frames is set as True or False accordingly."

REFERENCE
 "clause 12.18.2.2.2 b.9"
DEFVAL { false }
::= { ieee8021DdcfmrEntry 12 }

```
ieee8021DdcfmrFloodingFlag OBJECT-TYPE
    SYNTAX TruthValue
    MAX-ACCESS read-create
    STATUS current
    DESCRIPTION
        "True indicates that flooding is allowed if Egress port can't be
         identified for RFM by the Filtering Database, False otherwise."
    REFERENCE
        "clause 12.18.2.2.2 b.10"
    DEFVAL { true }
    ::= { ieee8021DdcfmrEntry 13 }
```

```
ieee8021DdcfmrTruncationFlag OBJECT-TYPE
    SYNTAX TruthValue
    MAX-ACCESS read-create
    STATUS current
    DESCRIPTION
        "True indicates that the received data frame will be truncated to
         keep the contracted RFM size not exceeding the egress port's
         Maximum Service Data Unit Size, False otherwise."
    REFERENCE
        "clause 12.18.2.2.2 b.11"
    DEFVAL { true }
    ::= { ieee8021DdcfmrEntry 14 }
```

```
ieee8021DdcfmrActivationStatus OBJECT-TYPE
    SYNTAX TruthValue
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "True When receiving a request to activate a Reflection Responder,
         False When receiving a request to stop Reflection Responder or
         its timer expires."
    REFERENCE
        "clause 12.18.2.3.3 b.12"
    DEFVAL { false }
    ::= { ieee8021DdcfmrEntry 15 }
```

```
ieee8021DdcfmrRemainDuration OBJECT-TYPE
    SYNTAX Unsigned32
    UNITS "seconds"
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "The value indicates the time, in the unit of seconds, or count
         left for Reflection Responder to be active."
    REFERENCE
        "clause 12.18.2.3.3 b.13"
    ::= { ieee8021DdcfmrEntry 16 }
```

```
ieee8021DdcfmrNextRfmTransID OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "It indicates the value of RFM Transaction Identifier field of the
         next RFM transmitted. It is incremented by 1 with each
         transmission of RFM."
```

```

REFERENCE
    "clause 12.18.2.3.3 b.14"
 ::= { ieee8021DdcfmrRrEntry 17 }

ieee8021DdcfmrRowStatus 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."
 ::= { ieee8021DdcfmrEntry 18 }

-- ****
-- The RFM Receiver Table
-- ****

ieee8021DdcfmrRFMReceiverTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021DdcfmrRFMReceiverEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The RFM Receiver MIB object table. Each row in the table
    represents a different RFM Receiver.
    All rows associated with this table persist across system restart."
REFERENCE
    "clauses 12.18.3"
 ::= { ieee8021DdcfmrRFMReceiver 1 }

ieee8021DdcfmrRFMReceiverEntry OBJECT-TYPE
SYNTAX Ieee8021DdcfmrRFMReceiverEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The DDCFM RFM Receiver. Each entry associated with a DDCFM RFM
    Receiver that reference to a MP."
INDEX { ieee8021DdcfmrFmReceiverIfIndex,
ieee8021DdcfmrFmReceiverMdIndex}
 ::= { ieee8021DdcfmrRFMReceiverTable 1 }

Ieee8021DdcfmrRFMReceiverEntry ::= SEQUENCE {
    ieee8021DdcfmrFmReceiverIfIndex InterfaceIndex,
    ieee8021DdcfmrFmReceiverMdIndex Dot1agCfmMDLevel,
    ieee8021DdcfmrRFMRowStatus RowStatus
}

ieee8021DdcfmrFmReceiverIfIndex OBJECT-TYPE
SYNTAX InterfaceIndex
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The interface index of the interface either a
    bridge port, or an aggregated IEEE802.3 port within a Bridge
    Port, on which the RFM Receiver is created.
    When the ifIndex value corresponds to the ifIndex of a
    Bridge Port, the value in this column must match the value in the
    ieee8021BridgeBasePortIfIndex column for the bridge port."
REFERENCE

```

```
        "clause 12.18.3.1.2 a.2 "
        ::= { ieee8021DdcfmrFMReceiverEntry 1 }

ieee8021DdcfmrFmReceiverMdIndex OBJECT-TYPE
    SYNTAX Dot1agCfmMDLevel
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "MD level of the RFM Receiver managed object."
    REFERENCE
        "clause 12.18.3.1.2 a.2"
        ::= { ieee8021DdcfmrFMReceiverEntry 2 }

ieee8021DdcfmrFmRowStatus 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."
    ::= { ieee8021DdcfmrFMReceiverEntry 3 }

-- *****
-- The Decapsulator Responder Table
-- *****

ieee8021DdcfmrDrTable OBJECT-TYPE
    SYNTAX SEQUENCE OF Ieee8021DdcfmrDrEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The DDCFM Decapsulator Responder MIB object table. Each row in the
        table represents a different DDCFM Decapsulator Responder. All rows
        in this table persist across a system restart; however after such
        a restart, the value of the ActivationStatus column will be false."
    REFERENCE
        "clauses 12.18.4"
    ::= { ieee8021DdcfmrDr 1 }

ieee8021DdcfmrDrEntry OBJECT-TYPE
    SYNTAX Ieee8021DdcfmrDrEntry
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The DDCFM Decapsulator Responder. Each entry associated with a
        DDCFM RFM Receiver which reference to a MP."
    INDEX { ieee8021DdcfmrIfIndex,
            ieee8021DdcfmrMdIndex,
            ieee8021DdcfmrVlanIdOrNone }
    ::= { ieee8021DdcfmrDrTable 1 }

Ieee8021DdcfmrDrEntry ::= SEQUENCE {
    ieee8021DdcfmrIfIndex InterfaceIndex,
    ieee8021DdcfmrMdIndex Dot1agCfmMDLevel,
    ieee8021DdcfmrVlanIdOrNone VlanIdOrNone,
    ieee8021DdcfmrSfmOriginator MacAddress,
    ieee8021DdcfmrSourceAddressStayFlag TruthValue,
    ieee8021DdcfmrFloodingFlag TruthValue,
    ieee8021DdcfmrDuration Unsigned32,
```

```

ieee8021DdcfmdrActivationStatus TruthValue,
ieee8021DdcfmdrRemainDuration Unsigned32,
ieee8021DdcfmdrSfmSequenceErrors Unsigned32,
ieee8021DdcfmdrRowStatus RowStatus
}

ieee8021DdcfmdrIfIndex OBJECT-TYPE
SYNTAX InterfaceIndex
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The interface index of the interface either a Bridge
Port, or an aggregated IEEE802.3 port within a Bridge
Port, on which the Decapsulator Responder is created.
When the ifIndex value corresponds to the ifIndex of a
Bridge Port, the value in this column must match the value in the
ieee8021BridgeBasePortIfIndex column for the bridge port."
REFERENCE
"clause 12.18.4.1.2 a.2"
 ::= { ieee8021DdcfmdrEntry 1 }

ieee8021DdcfmdrMdIndex OBJECT-TYPE
SYNTAX Dot1agCfmMDLevel
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"MD level of the Decapsulator Responder managed object."
REFERENCE
"clause 12.18.4.1.2 a.2"
 ::= { ieee8021DdcfmdrEntry 2 }

ieee8021DdcfmdrVlanIdOrNone OBJECT-TYPE
SYNTAX VlanIdOrNone
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
>An integer indicating the VID expected from the Send Frame Message
frames."
REFERENCE
"clause 12.18.4.1.2 a.2"
 ::= { ieee8021DdcfmdrEntry 3 }

ieee8021DdcfmdrSfmOriginator OBJECT-TYPE
SYNTAX MacAddress
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"MAC address reference to the corresponding Send Frame Message
Originator."
REFERENCE
"clause 12.18.4.2.3 b.2"
 ::= { ieee8021DdcfmdrEntry 4 }

ieee8021DdcfmdrSourceAddressStayFlag OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"True indicates that Decapsulator Responder does not replace the

```

```
        source_address field of the decapsulated frame with the
        Decapsulator Responder's own MAC address, False otherwise."
REFERENCE
    "clause 12.18.4.2.3 b.3"
DEFVAL { true }
 ::= { ieee8021DdcfmdrEntry 5 }

ieee8021DdcfmdrFloodingFlag OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
    "True indicates that broadcast is allowed if Egress port can't be
     identified by the Filtering Database, False otherwise."
REFERENCE
    "clause 12.18.4.3.2 b.3"
DEFVAL { true }
 ::= { ieee8021DdcfmdrEntry 6 }

ieee8021DdcfmdrDuration OBJECT-TYPE
SYNTAX Unsigned32
UNITS "seconds"
MAX-ACCESS read-create
STATUS current
DESCRIPTION
    "The time that the Decapsulator Responder can stay active after
     its activation in the unit of seconds."
REFERENCE
    "clause 12.18.4.3.2 b.4"
 ::= { ieee8021DdcfmdrEntry 7 }

ieee8021DdcfmdrActivationStatus OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "True When receiving a request to activate a Decapsulator
     Responder, false When receiving a request to stop the Decapsulator
     Responder or its timer expires."
REFERENCE
    "clause 12.18.4.2.3 b.6"
DEFVAL { false }
 ::= { ieee8021DdcfmdrEntry 8 }

ieee8021DdcfmdrRemainDuration OBJECT-TYPE
SYNTAX Unsigned32
UNITS "seconds"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "The value indicates the time left for Decapsulator Responder keep
     active. Its granularity is in seconds."
REFERENCE
    "clause 12.18.4.2.3 b.7"
 ::= { ieee8021DdcfmdrEntry 9 }

ieee8021DdcfmdrSfmSequenceErrors OBJECT-TYPE
SYNTAX Unsigned32
UNITS "integer"
```

```

MAX-ACCESS read-only
STATUS current
DESCRIPTION
    "The value indicates the total number of out-of-sequence SFMs."
REFERENCE
    "clause 12.18.4.2.3 b.8"
 ::= { ieee8021DdcfmdrEntry 10 }

ieee8021DdcfmdrRowStatus 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."
 ::= { ieee8021DdcfmdrEntry 11 }

-- *****
-- The SFM Originator Table
-- *****

ieee8021Ddcfmsotable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021Ddcfmsotentry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "The DDCFM Send Frame Message Originator MIB object table. Each row
    in the table represents a different DDCFM Send Frame Message
    Originator. All rows in this table persist across a system restart;
    however after such a restart, the value of the ActivationStatus
    column will be false."
REFERENCE
    "clauses 12.18.5"
 ::= { ieee8021Ddcfmsfmonitor 1 }

ieee8021Ddcfmsotentry OBJECT-TYPE
SYNTAX Ieee8021Ddcfmsotentry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "Each entry associated with a Send Frame Message Originator."
INDEX { ieee8021DdcfmsfmIfIndex,
        ieee8021DdcfmsotmdIndex,
        ieee8021DdcfmsovlanIdOrNone,
        ieee8021Ddcfmsodirection}
 ::= { ieee8021Ddcfmsotable 1 }

Ieee8021Ddcfmsotentry ::= SEQUENCE {
    ieee8021DdcfmsfmIfIndex InterfaceIndex,
    ieee8021DdcfmsotmdIndex Dot1agCfmMDLevel,
    ieee8021DdcfmsovlanIdOrNone VlanIdOrNone,
    ieee8021Ddcfmsodirection Dot1agCfmMpDirection,
    ieee8021DdcfmsodrMacAddress MacAddress,
    ieee8021Ddcfmsoduration Unsigned32,
    ieee8021Ddcfmsoactivationstatus TruthValue,
    ieee8021Ddcfmsoremainduration Unsigned32,
    ieee8021Ddcfmsorowstatus RowStatus
}

```

```
ieee8021DdcfmsfmIfIndex OBJECT-TYPE
    SYNTAX InterfaceIndex
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The interface index of the interface either a bridge
         port, or an aggregated IEEE802.3 port within a bridge
         port, on which Send Frame Message Originator is created.
         When the ifIndex value corresponds to the ifIndex of a
         Bridge Port, the value in this column must match the value in the
         ieee8021BridgeBasePortIfIndex column for the bridge port."
    REFERENCE
        "clause 12.18.5.1.2 a.2"
    ::= { ieee8021DdcfmsmEntry 1 }

ieee8021DdcfsmSoMdIndex OBJECT-TYPE
    SYNTAX Dot1agCfmMDIlevel
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "MD level of the Send Frame Message Originator managed object."
    REFERENCE
        "clause 12.18.5.1.2 a.2"
    ::= { ieee8021DdcfsmSoEntry 2 }

ieee8021DdcfsmSoVlanIdOrNone OBJECT-TYPE
    SYNTAX VlanIdOrNone
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "An integer indicating the VID to be used on Send Frame Message
         frames."
    REFERENCE
        "clause 12.18.5.1.2 a.2"
    ::= { ieee8021DdcfsmSoEntry 3 }

ieee8021DdcfsmSoDirection OBJECT-TYPE
    SYNTAX Dot1agCfmMpDirection
    MAX-ACCESS not-accessible
    STATUS current
    DESCRIPTION
        "The direction in which the SFM Originator faces."
    REFERENCE
        "clause 12.18.5.1.2 a.2"
    ::= { ieee8021DdcfsmSoEntry 4 }

ieee8021DdcfsmSoDrMacAddress OBJECT-TYPE
    SYNTAX MacAddress
    MAX-ACCESS read-create
    STATUS current
    DESCRIPTION
        "MAC Address of the corresponding Decapsulator Responder."
    REFERENCE
        "clause 12.18.5.4.2 b"
    ::= { ieee8021DdcfsmSoEntry 5 }

ieee8021DdcfsmSoDuration OBJECT-TYPE
    SYNTAX Unsigned32
    UNITS "seconds"
```

```

MAX-ACCESS read-create
STATUS current
DESCRIPTION
  "Duration, in the unit of seconds, of Send Frame Message Originator
   staying active once activated."
REFERENCE
  "clause 12.18.5.4.2 c"
 ::= { ieee8021DdcfSoEntry 6 }

ieee8021DdcfSoActivationStatus OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
  "True When receiving a request to activate a Send Frame Message
   Originator, false When receiving a request to stop the Send Frame
   Message Originator or its timer expires."
REFERENCE
  "clause 12.18.5.2.3 b.4"
DEFVAL { false }
 ::= { ieee8021DdcfSoEntry 7 }

ieee8021DdcfSoRemainDuration OBJECT-TYPE
SYNTAX Unsigned32
UNITS "seconds"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
  "The value indicates the time left for Send Frame Message
   Originator keep active. Its granularity is in seconds."
REFERENCE
  "clause 12.18.5.2.3 b.5"
 ::= { ieee8021DdcfSoEntry 8 }

ieee8021DdcfSoRowStatus 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."
 ::= { ieee8021DdcfSoEntry 9 }

-- *****
-- IEEE802.1Qaw MIB Module - Conformance Information
-- *****
ieee8021DdcfCompliances OBJECT IDENTIFIER ::= { ieee8021DdcfConformance 1 }
ieee8021DdcfGroups OBJECT IDENTIFIER ::= { ieee8021DdcfConformance 2 }

-- *****
-- Units of conformance
-- *****
ieee8021DdcfStackGroup OBJECT-GROUP
OBJECTS {
  ieee8021DdcfStackRrMdLevel,
  ieee8021DdcfStackRrDirection,
  ieee8021DdcfStackRFMreceiverMdLevel,
}

```

```
ieee8021DdcfStackDrMdLevel,
ieee8021DdcfStackDrVlanIdOrNone,
ieee8021DdcfStackSFMOriginatorMdLevel,
ieee8021DdcfStackSFMOriginatorVlanIdOrNone,
ieee8021DdcfStackSFMOriginatorDirection
}
STATUS current
DESCRIPTION
"Objects for the DDCFM Stack group."
::= { ieee8021DdcfGroups 1 }

ieee8021DdcfRrGroup OBJECT-GROUP
OBJECTS {
    ieee8021DdcfRrPrimaryVlanIdOrNone,
    ieee8021DdcfRrFilter,
    ieee8021DdcfRrSamplingInterval,
    ieee8021DdcfRrTargetAddress,
    ieee8021DdcfRrContinueFlag,
    ieee8021DdcfRrDuration,
    ieee8021DdcfRrDurationInTimeFlag,
    ieee8021DdcfRrVlanPriority,
    ieee8021DdcfRrVlanDropEligible,
    ieee8021DdcfRrFloodingFlag,
    ieee8021DdcfRrTruncationFlag,
    ieee8021DdcfRrActivationStatus,
    ieee8021DdcfRrRemainDuration,
    ieee8021DdcfRrNextRfmTransID,
    ieee8021DdcfRrRowStatus
}
STATUS current
DESCRIPTION
"Objects for the Reflection Responder group."
::= { ieee8021DdcfGroups 2 }

ieee8021DdcfRFMReceiverGroup OBJECT-GROUP
OBJECTS {
    ieee8021DdcfRFMRowStatus
}
STATUS current
DESCRIPTION
"Objects for the RFM Receiver group."
::= { ieee8021DdcfGroups 3 }

ieee8021DdcfDrGroup OBJECT-GROUP
OBJECTS {
    ieee8021DdcfDrSourceAddressStayFlag,
    ieee8021DdcfDrSfmOriginator,
    ieee8021DdcfDrFloodingFlag,
    ieee8021DdcfDrDuration,
    ieee8021DdcfDrActivationStatus,
    ieee8021DdcfDrRemainDuration,
    ieee8021DdcfDrSFMsequenceErrors,
    ieee8021DdcfDrRowStatus
}
STATUS current
DESCRIPTION
"Objects for the Decapsulator Responder group."
::= { ieee8021DdcfGroups 4 }
```

```

ieee8021DdcfSoGroup OBJECT-GROUP
OBJECTS {
    ieee8021DdcfSoDrMacAddress,
    ieee8021DdcfSoDuration,
    ieee8021DdcfSoActivationStatus,
    ieee8021DdcfSoRemainDuration,
    ieee8021DdcfSoRowStatus
}
STATUS current
DESCRIPTION
    "Objects for the Send Frame Message Originator group."
::= { ieee8021DdcfGroups 5 }

--*****MIB Module Compliance statements*****
--*****MODULE-COMPLIANCE*****
ieee8021DdcfCompliance MODULE-COMPLIANCE
STATUS current
DESCRIPTION
    "The compliance statement for support of the DDCFM MIB module."
MODULE
MANDATORY-GROUPS {
    ieee8021DdcfStackGroup,
    ieee8021DdcfRrGroup,
    ieee8021DdcfRFMReceiverGroup,
    ieee8021DdcfDrGroup,
    ieee8021DdcfSoGroup
}

OBJECT ieee8021DdcfRrRowStatus
SYNTAX RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
    destroy(6) }
DESCRIPTION "Support for createAndWait is not required."

OBJECT ieee8021DdcfRFMRowStatus
SYNTAX RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
    destroy(6) }
DESCRIPTION "Support for createAndWait is not required."

OBJECT ieee8021DdcfDrRowStatus
SYNTAX RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
    destroy(6) }
DESCRIPTION "Support for createAndWait is not required."

OBJECT ieee8021DdcfSoRowStatus
SYNTAX RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
    destroy(6) }
DESCRIPTION "Support for createAndWait is not required."
::= { ieee8021DdcfCompliances 1 }

END

```

17.7.10 Definitions for the IEEE8021-PBBTE MIB module

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

IMPORTS
    MODULE-IDENTITY,
    NOTIFICATION-TYPE,
    OBJECT-TYPE,
    Unsigned32
        FROM SNMPv2-SMI
    RowStatus,
    StorageType,
    TruthValue
        FROM SNMPv2-TC
    ieee802dot1mibs,
    IEEE8021BridgePortNumber,
    IEEE8021PbbComponentIdentifier,
    IEEE8021PbbServiceIdentifier,
    IEEE8021PbbTeEsp,
    IEEE8021PbbTeProtectionGroupActiveRequests,
    IEEE8021PbbTeProtectionGroupId,
    IEEE8021PbbTeProtectionGroupConfigAdmin,
    IEEE8021PbbTeTSidId,
    IEEE8021VlanIndexOrWildcard
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBaseComponentId
        FROM IEEE8021-BRIDGE-MIB
    PortList
        FROM Q-BRIDGE-MIB
    ieee8021QBridgeVlanCurrentComponentId
        FROM IEEE8021-Q-BRIDGE-MIB
    MODULE-COMPLIANCE,
    NOTIFICATION-GROUP,
    OBJECT-GROUP
        FROM SNMPv2-CONF;

ieee8021PbbTeMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
ORGANIZATION "IEEE 802.1 Working Group"
CONTACT-INFO
    " WG-URL: http://grouper.ieee.org/groups/8021/1/index.html
    WG-EMAIL: stds-802-1@ieee.org

    Contact: Kevin Nolish
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG

    Contact: David Levi
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
```

Piscataway
NJ 08855-1331
USA
E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"

DESCRIPTION

"Copyright (C) IEEE. All Rights Reserved
This version of this MIB module is part of IEEE 802.1Q;
See the standard itself for full legal notices.

Unless otherwise indicated, the references in this MIB
module are to IEEE 802.1Q-2011"

REVISION "201102270000Z" -- February 27, 2011

DESCRIPTION

"Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."

REVISION "200811180000Z" -- November 18, 2008

DESCRIPTION

"Initial version of the PBB-TE MIB module based upon draft 3.2
of the MIB modules defined in 802.1ap and 802.1Qay"

::= { ieee802dot1mibs 10 }

ieee8021PbbTeNotifications OBJECT IDENTIFIER ::= { ieee8021PbbTeMib 0 }
ieee8021PbbTeObjects OBJECT IDENTIFIER ::= { ieee8021PbbTeMib 1 }
ieee8021PbbTeConformance OBJECT IDENTIFIER ::= { ieee8021PbbTeMib 2 }

--

-- 802.1Qay MIB Objects

--

ieee8021PbbTeProtectionGroupListTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021PbbTeProtectionGroupListEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The PBB-TE Protection group list table. Each entry in this table
corresponds to a configured PBB-TE Protection Group configured on
the B-Component of an IB-BEB."
REFERENCE
"12.19.1"
 ::= { ieee8021PbbTeObjects 1 }

ieee8021PbbTeProtectionGroupListEntry OBJECT-TYPE
SYNTAX Ieee8021PbbTeProtectionGroupListEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The PBB-TE protection group list table entry.
Note that the component ID must refer to an B component"
INDEX { ieee8021BridgeBaseComponentId,
ieee8021PbbTeProtectionGroupListGroupId }
 ::= { ieee8021PbbTeProtectionGroupListTable 1 }

Ieee8021PbbTeProtectionGroupListEntry ::=
SEQUENCE {

```
ieee8021PbbTeProtectionGroupId OBJECT-TYPE
  IEEE8021PbbTeProtectionGroupId,
    ieee8021PbbTeProtectionGroupListMD          Unsigned32,
    ieee8021PbbTeProtectionGroupListWorkingMA   Unsigned32,
    ieee8021PbbTeProtectionGroupListProtectionMA Unsigned32,
    ieee8021PbbTeProtectionGroupListStorageType  StorageType,
    ieee8021PbbTeProtectionGroupListRowStatus     RowStatus
  }

ieee8021PbbTeProtectionGroupListGroupId OBJECT-TYPE
  SYNTAX      IEEE8021PbbTeProtectionGroupId
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The unique identifier for the protection group."
  REFERENCE
    "12.19.2"
  ::= { ieee8021PbbTeProtectionGroupListEntry 1 }

ieee8021PbbTeProtectionGroupListMD OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This refers to the Maintenance Domain that qualifies the
     WorkingMA and ProtectionMA columns of this table. The MD
     index in this column must hold a value that matches the
     in the dotlagCfmStackMdIndex in the dotlagCfmStackTable
     for the corresponding WorkingMA and ProtectionMA columns
     of this table. This correspondence must hold for the RowStatus
     of this row to be set to Active. Furthermore, this column may
     not be modified while the RowStatus for this row is Active"
  REFERENCE
    "12.19.1.1.3 b)"
  ::= { ieee8021PbbTeProtectionGroupListEntry 2 }

ieee8021PbbTeProtectionGroupListWorkingMA OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This refers to the Maintenance Association that refers
     to the PBB-TE MA that corresponds to the group's working
     entity. The MA index in this column must hold a value
     that is the value of dotlagCfmStackMaIndex column for
     some entry in the dotlagCfmStackTable before the RowStatus
     for this row can be set to Active. Furthermore, this column
     may not be modified when the RowStatus for this row is Active."
  REFERENCE
    "12.19.1.1.3 b)"
  ::= { ieee8021PbbTeProtectionGroupListEntry 3 }

ieee8021PbbTeProtectionGroupListProtectionMA OBJECT-TYPE
  SYNTAX      Unsigned32
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This refers to the Maintenance Association that refers
```

to the PBB-TE MA that corresponds to the group's protection entity. The MA index in this column must hold a value that is the value of dot1agCfmStackMaIndex column for some entry in the dot1agCfmStackTable before the RowStatus for this row can be set to Active. Furthermore, this column may not be modified when the RowStatus for this row is Active."

REFERENCE

"12.19.1.1.3 c)"

::= { ieee8021PbbTeProtectionGroupListEntry 4 }

ieee8021PbbTeProtectionGroupListStorageType OBJECT-TYPE

SYNTAX StorageType

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"This object indicates the persistence of this entry. All read-create columns must be writable if this column is set to permanent."

DEFVAL { nonVolatile }

::= { ieee8021PbbTeProtectionGroupListEntry 5 }

ieee8021PbbTeProtectionGroupListRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The status of this row.

The writable columns in a row cannot be changed if the row is active. The PbbTeProtectionGroupListWorkingMA, and PbbTeProtectionGroupListProtectionMA columns must be specified before the row can be activated."

REFERENCE

"12.19.1.2"

::= { ieee8021PbbTeProtectionGroupListEntry 6 }

ieee8021PbbTeMASHaredGroupTable OBJECT-TYPE

SYNTAX SEQUENCE OF Ieee8021PbbTeMASHaredGroupEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table contains references to all protection groups that share a working or protection entity with a given protection group."

REFERENCE

"12.19.1.1.3 d)"

::= { ieee8021PbbTeObjects 2 }

ieee8021PbbTeMASHaredGroupEntry OBJECT-TYPE

SYNTAX Ieee8021PbbTeMASHaredGroupEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The ieee8021PbbTeMASHaredGroupEntry table. The table is indexed by protection group and by a simple integer. The table lists all protection groups that load share with that group."

INDEX { ieee8021BridgeBaseComponentId,
ieee8021PbbTeProtectionGroupListGroupId,
ieee8021PbbTeMASHaredGroupSubIndex }

```
 ::= { ieee8021PbbTeMASHaredGroupTable 1 }

Ieee8021PbbTeMASHaredGroupEntry ::=  
SEQUENCE {  
    ieee8021PbbTeMASHaredGroupSubIndex Unsigned32,  
    ieee8021PbbTeMASHaredGroupId      IEEE8021PbbTeProtectionGroupId  
}

ieee8021PbbTeMASHaredGroupSubIndex OBJECT-TYPE  
SYNTAX Unsigned32(1..4294967295)  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"A simple integer to distinguish the members of the set  
of MAs that comprise the set of load sharing MAs for the  
specified protection group."  
 ::= { ieee8021PbbTeMASHaredGroupEntry 1 }

ieee8021PbbTeMASHaredGroupId OBJECT-TYPE  
SYNTAX IEEE8021PbbTeProtectionGroupId  
MAX-ACCESS read-only  
STATUS current  
DESCRIPTION  
"This column holds the group id of a protection group that shares  
a working or protection group with the group whose index is the  
first index of this row."  
 ::= { ieee8021PbbTeMASHaredGroupEntry 2 }

ieee8021PbbTeTesiTable OBJECT-TYPE  
SYNTAX SEQUENCE OF Ieee8021PbbTeTesiEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"The PBB-TE TESI table contains information for each  
TE Service Instance known to a system."  
REFERENCE  
"12.16.5.3.1"  
 ::= { ieee8021PbbTeObjects 3 }

ieee8021PbbTeTesiEntry OBJECT-TYPE  
SYNTAX Ieee8021PbbTeTesiEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
"The PBB-TE TESI entry. Each entry maps a TESI to  
a component and CBP."  
INDEX { ieee8021PbbTeTesiId }  
 ::= { ieee8021PbbTeTesiTable 1 }

Ieee8021PbbTeTesiEntry ::=  
SEQUENCE {  
    ieee8021PbbTeTesiId          IEEE8021PbbTeTSidId,  
    ieee8021PbbTeTesiComponent   IEEE8021PbbComponentIdentifier,  
    ieee8021PbbTeTesiBridgePort  IEEE8021BridgePortNumber,  
    ieee8021PbbTeTesiStorageType StorageType,  
    ieee8021PbbTeTesiRowStatus   RowStatus  
}
```

```

ieee8021PbbTeTesiId OBJECT-TYPE
  SYNTAX      IEEE8021PbbTeTSidId
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "This is the unique identifier for a PBB-TE TE-SID."
  REFERENCE
    "802.1Qay 3.11"
  ::= { ieee8021PbbTeTesiEntry 1 }

ieee8021PbbTeTesiComponent OBJECT-TYPE
  SYNTAX      IEEE8021PbbComponentIdentifier
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This is the component upon which the bridge port of the
     TESI is located."
  REFERENCE
    "12.16.5.3.2 a)"
  ::= { ieee8021PbbTeTesiEntry 2 }

ieee8021PbbTeTesiBridgePort OBJECT-TYPE
  SYNTAX      IEEE8021BridgePortNumber
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This is the bridge port of the TESI."
  REFERENCE
    "12.16.5.3.2 b)"
  ::= { ieee8021PbbTeTesiEntry 3 }

ieee8021PbbTeTesiStorageType OBJECT-TYPE
  SYNTAX      StorageType
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This object indicates the persistence of this entry. All read-create
     columns must be writable for rows whose StorageType is permanent."
  DEFVAL { nonVolatile }
  ::= { ieee8021PbbTeTesiEntry 4 }

ieee8021PbbTeTesiRowStatus OBJECT-TYPE
  SYNTAX      RowStatus
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This column holds the status for this row.

    When the status is active, no columns of this table may be
    modified. All columns must have a valid value before the row
    can be activated."
  ::= { ieee8021PbbTeTesiEntry 5 }

ieee8021PbbTeTeSiEspTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF Ieee8021PbbTeTeSiEspEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The PBB-TE TE-SID table contains information for each TE

```

```
        service instance known to a system."
REFERENCE
    "12.16.5.3.2 c)"
 ::= { ieee8021PbbTeObjects 4 }

ieee8021PbbTeTeSiEspEntry OBJECT-TYPE
SYNTAX      Ieee8021PbbTeTeSiEspEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The PBB-TE TE-SID entry. Each entry refers to a
     TE Service Instance by identifier and contains information about
     one of the ESPs that comprise this TE Service Instance."
INDEX { ieee8021PbbTeTesiId,
         ieee8021PbbTeTeSiEspEspIndex }
 ::= { ieee8021PbbTeTeSiEspTable 1 }

Ieee8021PbbTeTeSiEspEntry ::=*
SEQUENCE {
    ieee8021PbbTeTeSiEspEspIndex      Unsigned32,
    ieee8021PbbTeTeSiEspEsp          IEEE8021PbbTeEsp,
    ieee8021PbbTeTeSiEspStorageType  StorageType,
    ieee8021PbbTeTeSiEspRowStatus   RowStatus
}

ieee8021PbbTeTeSiEspEspIndex OBJECT-TYPE
SYNTAX      Unsigned32 (1..4294967295)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "This is an identifier, of local significance to a particular
     PBB-TE TE-SID which is used to index all of the ESPs associated
     with the TE-SID."
REFERENCE
    "12.16.5.3.2 c)"
 ::= { ieee8021PbbTeTeSiEspEntry 1 }

ieee8021PbbTeTeSiEspEsp OBJECT-TYPE
SYNTAX      IEEE8021PbbTeEsp
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This column holds the ESP identifier for one of the Ethernet
     Switched Paths that define the TE service instance."
REFERENCE
    "12.16.5.3.2 c)"
 ::= { ieee8021PbbTeTeSiEspEntry 2 }

ieee8021PbbTeTeSiEspStorageType OBJECT-TYPE
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object indicates the persistence of this entry. All read-create
     columns must be writable for permanent rows."
DEFVAL { nonVolatile }
 ::= { ieee8021PbbTeTeSiEspEntry 3 }

ieee8021PbbTeTeSiEspRowStatus OBJECT-TYPE
```

```

SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This column holds the status for this row.

When the status is active, no columns of this table may be
modified. All columns must have a valid value before the row
can be activated."
 ::= { ieee8021PbbTeTeSiEspEntry 4 }

ieee8021PbbTeProtectionGroupConfigTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021PbbTeProtectionGroupConfigEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION

    "The PBB-TE Protection group config table contains
    configuration and status information for each configuration
    group configured in the system. Entries in this table are
    created implicitly by the creation of entries in the
    ieee8021PbbTeProtectionGroupListTable table."
REFERENCE
    "12.19.2"
 ::= { ieee8021PbbTeObjects 5 }

ieee8021PbbTeProtectionGroupConfigEntry OBJECT-TYPE
SYNTAX      Ieee8021PbbTeProtectionGroupConfigEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The protection group configuration table entry. Rows are
    created in this table implicitly when a row is added to the
    ieee8021PbbTeProtectionGroupListTable."
INDEX { ieee8021BridgeBaseComponentId,
        ieee8021PbbTeProtectionGroupListGroupId }
 ::= { ieee8021PbbTeProtectionGroupConfigTable 1 }

Ieee8021PbbTeProtectionGroupConfigEntry ::=

SEQUENCE {
    ieee8021PbbTeProtectionGroupConfigState
        INTEGER,
    ieee8021PbbTeProtectionGroupConfigCommandStatus
        IEEE8021PbbTeProtectionGroupConfigAdmin,
    ieee8021PbbTeProtectionGroupConfigCommandLast
        IEEE8021PbbTeProtectionGroupConfigAdmin,
    ieee8021PbbTeProtectionGroupConfigCommandAdmin
        IEEE8021PbbTeProtectionGroupConfigAdmin,
    ieee8021PbbTeProtectionGroupConfigActiveRequests
        IEEE8021PbbTeProtectionGroupActiveRequests,
    ieee8021PbbTeProtectionGroupConfigWTR
        Unsigned32,
    ieee8021PbbTeProtectionGroupConfigHoldOff
        Unsigned32,
    ieee8021PbbTeProtectionGroupConfigNotifyEnable
        TruthValue,
    ieee8021PbbTeProtectionGroupConfigStorageType
        StorageType
}

```

```
ieee8021PbbTeProtectionGroupConfigState OBJECT-TYPE
    SYNTAX    INTEGER {
        workingPath(1),
        protectionPat(2),
        waitToRestore(3),
        protAdmin(4)
    }
    MAX-ACCESS  read-only
    STATUS     current
    DESCRIPTION
        "This column indicates the current state of the protection
         switching state machine for a protection group."
    REFERENCE "26.10.3.5 12.19.2.1.3 c)"
    ::= { ieee8021PbbTeProtectionGroupConfigEntry 1 }

ieee8021PbbTeProtectionGroupConfigCommandStatus OBJECT-TYPE
    SYNTAX    IEEE8021PbbTeProtectionGroupConfigAdmin
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "This column indicates the status of administrative commands
         within the protection group. It reflects the current
         operational administrative command being acted upon by the
         protection group."
    REFERENCE "12.19.2.1.3 d)"
    ::= { ieee8021PbbTeProtectionGroupConfigEntry 2 }

ieee8021PbbTeProtectionGroupConfigCommandLast OBJECT-TYPE
    SYNTAX    IEEE8021PbbTeProtectionGroupConfigAdmin
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
        "This column indicates the last attempted administrative
         command applied to the protection group. It is changed
         whenever a write is made to the CommandAdmin column of
         this table and is a record of the last attempted
         administrative operation."
    REFERENCE "12.19.2.1.3 d)"
    ::= { ieee8021PbbTeProtectionGroupConfigEntry 3 }

ieee8021PbbTeProtectionGroupConfigCommandAdmin OBJECT-TYPE
    SYNTAX    IEEE8021PbbTeProtectionGroupConfigAdmin
    MAX-ACCESS read-create
    STATUS     current
    DESCRIPTION
        "This column is used by the operator to request that the
         protection group state machine perform some administrative
         operation. The operator requests a command by writing the
         command value to this column. The state machine indicates the
         command that it is performing by setting the value of the
         CommandStatus column of this table. This column always reads
         back as clear(1)."
    REFERENCE "12.19.2.3.2"
    DEFVAL { clear }
    ::= { ieee8021PbbTeProtectionGroupConfigEntry 4 }

ieee8021PbbTeProtectionGroupConfigActiveRequests OBJECT-TYPE
    SYNTAX    IEEE8021PbbTeProtectionGroupActiveRequests
```

```
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This column shows the status of active requests within
the TE protection group."
REFERENCE "12.19.2.1.3 d)"
 ::= { ieee8021PbbTeProtectionGroupConfigEntry 5 }

ieee8021PbbTeProtectionGroupConfigWTR OBJECT-TYPE
SYNTAX Unsigned32 ( 0 | 5..12 )
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"This column is used to configure the wait-to-restore timer
for the protection group operation. The timer may be
configured in steps of 1 minute between 5 and 12 minutes, the
default being 5. Additionally, the value 0 is used to
indicate that the protection group is to operate
non-revertively."
REFERENCE "26.10.3.3.8 12.19.2.1.3 e)"
DEFVAL { 5 }
 ::= { ieee8021PbbTeProtectionGroupConfigEntry 6 }

ieee8021PbbTeProtectionGroupConfigHoldOff OBJECT-TYPE
SYNTAX Unsigned32( 0..100 )
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"This column is used to configure the hold off
timer. The purpose is to allow a service layer protection
mechanism to have a chance to fix the problem before
switching at the client layer, or to allow an upstream
protected domain to switch before a downstream domain. The
hold off timer has a period of from 0 to 10 seconds, the
default being 0, with a 100ms granularity."
REFERENCE "12.19.2.1.3 f)"
DEFVAL { 0 }
 ::= { ieee8021PbbTeProtectionGroupConfigEntry 7 }

ieee8021PbbTeProtectionGroupConfigNotifyEnable OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"This column is used to enable or disable transmission
of ieee8021PbbTeProtectionGroupAdminFailure notifications.
These notifications are generated whenever an administrative
command cannot be performed by the protection group."
DEFVAL { false }
 ::= { ieee8021PbbTeProtectionGroupConfigEntry 8 }

ieee8021PbbTeProtectionGroupConfigStorageType OBJECT-TYPE
SYNTAX StorageType
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"This object indicates the persistence of this entry. For
permanent objects the
ieee8021PbbTeProtectionGroupConfigCommandAdmin column
```

```
        must be writable."  
  
DEFVAL { nonVolatile }  
 ::= { ieee8021PbbTeProtectionGroupConfigEntry 9 }  
  
ieee8021PbbTeProtectionGroupISidTable OBJECT-TYPE  
SYNTAX SEQUENCE OF Ieee8021PbbTeProtectionGroupISidEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
    "This table contains entries for each I-SID that is  
    transported over TE-SIDs that belong to protection groups.  
    Each I-SID maps to a single protection group."  
REFERENCE "12.19.2.1.3 b)"  
 ::= { ieee8021PbbTeObjects 6 }  
  
ieee8021PbbTeProtectionGroupISidEntry OBJECT-TYPE  
SYNTAX Ieee8021PbbTeProtectionGroupISidEntry  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
    "The ieee8021PbbTeProtectionGroupISidTable entry."  
INDEX { ieee8021PbbTeProtectionGroupISidIndex }  
 ::= { ieee8021PbbTeProtectionGroupISidTable 1 }  
  
Ieee8021PbbTeProtectionGroupISidEntry ::=  
SEQUENCE {  
    ieee8021PbbTeProtectionGroupISidIndex      IEEE8021PbbServiceIdentifier,  
                                                ieee8021PbbTeProtectionGroupISidComponentId  
    IEEE8021PbbComponentIdentifier,  
                                                ieee8021PbbTeProtectionGroupISidGroupId  
    IEEE8021PbbTeProtectionGroupId,  
    ieee8021PbbTeProtectionGroupISidStorageType StorageType,  
    ieee8021PbbTeProtectionGroupISidRowStatus   RowStatus  
}  
  
ieee8021PbbTeProtectionGroupISidIndex OBJECT-TYPE  
SYNTAX IEEE8021PbbServiceIdentifier  
MAX-ACCESS not-accessible  
STATUS current  
DESCRIPTION  
    "This is the I-Sid that is to be mapped to a protection  
    group."  
 ::= { ieee8021PbbTeProtectionGroupISidEntry 1 }  
  
ieee8021PbbTeProtectionGroupISidComponentId OBJECT-TYPE  
SYNTAX IEEE8021PbbComponentIdentifier  
MAX-ACCESS read-create  
STATUS current  
DESCRIPTION  
    "This column qualifies the GroupId column to a particular  
    component."  
 ::= { ieee8021PbbTeProtectionGroupISidEntry 2 }  
  
ieee8021PbbTeProtectionGroupISidGroupId OBJECT-TYPE  
SYNTAX IEEE8021PbbTeProtectionGroupId  
MAX-ACCESS read-create  
STATUS current  
DESCRIPTION
```

```

    "This column contains the Id of the protection group used
    to transport the data belonging to the service identified
    by the I-SID value specified in the ISidIndex column of this
    table."
 ::= { ieee8021PbbTeProtectionGroupISidEntry 3 }

ieee8021PbbTeProtectionGroupISidStorageType OBJECT-TYPE
SYNTAX      StorageType
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This object indicates the persistence of this entry."
DEFVAL { nonVolatile }
 ::= { ieee8021PbbTeProtectionGroupISidEntry 4 }

ieee8021PbbTeProtectionGroupISidRowStatus OBJECT-TYPE
SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "This column contains the status for this row. Once active
    none of the columns in the row may be modified. All columns
    must be specified when creating the row."
 ::= { ieee8021PbbTeProtectionGroupISidEntry 5 }

ieee8021PbbTeBridgeStaticForwardAnyUnicastTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "A table containing forwarding information for each vlan
    specifying the set of ports to which forwarding of unicast
    addressed frames for which no more specific forwarding information
    applies. This is configured statically by management."
REFERENCE "8.8.1"
 ::= { ieee8021PbbTeObjects 7 }

ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry OBJECT-TYPE
SYNTAX Ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
    "Forwarding information for a VLAN, specifying the
    set of ports to which forwarding of unicast addressed
    frames for which no more specific forwarding information
    applies.

    The EgressPorts and ForbiddenPorts PortList objects, together,
    implement the PortMap control element listed in 802.1Q 8.8.1.c."
INDEX { ieee8021QBridgeVlanCurrentComponentId,
        ieee8021PbbTeBridgeStaticForwardAnyUnicastVlanIndex }
 ::= { ieee8021PbbTeBridgeStaticForwardAnyUnicastTable 1 }

Ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry ::=
SEQUENCE {
    ieee8021PbbTeBridgeStaticForwardAnyUnicastVlanIndex
    IEEE8021VlanIndexOrWildcard,
    ieee8021PbbTeBridgeStaticForwardAnyUnicastEgressPorts
}

```

```
        PortList,
ieee8021PbbTeBridgeStaticForwardAnyUnicastForbiddenPorts
        PortList,
ieee8021PbbTeBridgeStaticForwardAnyUnicastStorageType
        StorageType,
ieee8021PbbTeBridgeStaticForwardAnyUnicastRowStatus
        RowStatus
}

ieee8021PbbTeBridgeStaticForwardAnyUnicastVlanIndex OBJECT-TYPE
SYNTAX IEEE8021VlanIndexOrWildcard
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The VLAN-ID or other identifier referring to the VLAN to
which this static filtering entry applies."
 ::= { ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry 1 }

ieee8021PbbTeBridgeStaticForwardAnyUnicastEgressPorts OBJECT-TYPE
SYNTAX PortList
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The complete set of ports to which a unicast addressed frame
is to be forwarded. This value is persistent and will be restored
upon system reboot. A port may not be added to this set if it
is already a member of
ieee8021PbbTeBridgeStaticForwardAnyUnicastForbiddenPorts. The default
value is a string of zeros of appropriate length.

The value of this object MUST be retained across
reinitialization of the management system."
REFERENCE "8.8.1 c)"
 ::= { ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry 2 }

ieee8021PbbTeBridgeStaticForwardAnyUnicastForbiddenPorts OBJECT-TYPE
SYNTAX PortList
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The complete set of ports to which a unicast addressed frame
is to be filtered. This value is persistent and will be restored
upon system reboot. A port may not be added to this set if it
is already a member of
ieee8021PbbTeBridgeStaticForwardAnyUnicastEgress. The default
value is a string of zeros of appropriate length.

The value of this object MUST be retained across
reinitialization of the management system."
REFERENCE "8.8.1 c)"
 ::= { ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry 3 }

ieee8021PbbTeBridgeStaticForwardAnyUnicastStorageType OBJECT-TYPE
SYNTAX StorageType
MAX-ACCESS read-create
STATUS current
DESCRIPTION
"The storage type for this row. All read-create columns must be
writable for permanent entries."
```

```

DEFVAL { nonVolatile }
 ::= { ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry 4 }

ieee8021PbbTeBridgeStaticForwardAnyUnicastRowStatus OBJECT-TYPE
  SYNTAX      RowStatus
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
    "This column contains the status for this row."
 ::= { ieee8021PbbTeBridgeStaticForwardAnyUnicastEntry 5 }

-- ****
-- NOTIFICATIONS (TRAPS)
-- These notifications will be sent to the management entity
-- whenever a ProtectionGroup admin command cannot be performed
-- ****

ieee8021PbbTeProtectionGroupAdminFailure NOTIFICATION-TYPE
  OBJECTS {
    ieee8021PbbTeProtectionGroupConfigState,
    ieee8021PbbTeProtectionGroupConfigCommandStatus,
    ieee8021PbbTeProtectionGroupConfigCommandLast
  }
  STATUS  current
  DESCRIPTION
    "A protection group generates this notification whenever
     an administrative command cannot be executed by the
     protection state machine. For example, a requested
     manual switch cannot be performed because of a signal
     failure condition.

    The management entity receiving the notification can identify
    the system from the network source address of the notification
    and can identify the protection group by the indices of the OID
    of the ieee8021PbbTeProtectionGroupConfigState variable in the
    notification:

    ieee8021BridgeBaseComponentId - Identifies the
      component on the bridge where the protection group
      is configured.

    ieee8021PbbTeProtectionGroupListGroupId - The id
      of the protection group.

    "
 ::= { ieee8021PbbTeNotifications 1 }

--
-- MIB Module Compliance Statements
--
-- EDITOR's NOTE
-- These module compliance statements are simply a first cut. Mainly
-- they are here to keep smilint quiet.

ieee8021PbbTeCompliances OBJECT IDENTIFIER ::= { ieee8021PbbTeConformance 1 }
ieee8021PbbTeGroups      OBJECT IDENTIFIER ::= { ieee8021PbbTeConformance 2 }

--
-- Units of Conformance

```

```
ieee8021PbbTeGroupListGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeProtectionGroupListMD,
    ieee8021PbbTeProtectionGroupListWorkingMA,
    ieee8021PbbTeProtectionGroupListProtectionMA,
    ieee8021PbbTeProtectionGroupListStorageType,
    ieee8021PbbTeProtectionGroupListRowStatus
}
STATUS current
DESCRIPTION
    "Objects for the GroupList group."
::= { ieee8021PbbTeGroups 1 }

ieee8021PbbTeMAMSharedGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeMAMSharedGroupId
}
STATUS current
DESCRIPTION
    "Objects for the MA Load Sharing Table Group."
::= { ieee8021PbbTeGroups 2 }

ieee8021PbbTeTesiGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeTesiComponent,
    ieee8021PbbTeTesiBridgePort,
    ieee8021PbbTeTesiStorageType,
    ieee8021PbbTeTesiRowStatus
}
STATUS current
DESCRIPTION
    "Objects for the TE SI group"
::= { ieee8021PbbTeGroups 3 }

ieee8021PbbTeSiEspGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeTeSiEspEsp,
    ieee8021PbbTeTeSiEspStorageType,
    ieee8021PbbTeTeSiEspRowStatus
}
STATUS current
DESCRIPTION
    "Objects for the TESI ESP group."
::= { ieee8021PbbTeGroups 4 }

ieee8021PbbTeProtectionConfigManGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeProtectionGroupConfigState,
    ieee8021PbbTeProtectionGroupConfigCommandStatus,
    ieee8021PbbTeProtectionGroupConfigCommandLast,
    ieee8021PbbTeProtectionGroupConfigCommandAdmin,
    ieee8021PbbTeProtectionGroupConfigActiveRequests,
    ieee8021PbbTeProtectionGroupConfigNotifyEnable,
    ieee8021PbbTeProtectionGroupConfigStorageType
}
STATUS current
DESCRIPTION
    "Objects for the PbbTeConfiguration group."
```

```

 ::= { ieee8021PbbTeGroups 5 }

ieee8021PbbTeProtectionConfigOptGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeProtectionGroupConfigWTR,
    ieee8021PbbTeProtectionGroupConfigHoldOff
}
STATUS current
DESCRIPTION
    "Objects for the PbbTeConfiguration group."
 ::= { ieee8021PbbTeGroups 6 }

ieee8021PbbTeProtectionGroupISidGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeProtectionGroupISidComponentId,
    ieee8021PbbTeProtectionGroupISidGroupId,
    ieee8021PbbTeProtectionGroupISidStorageType,
    ieee8021PbbTeProtectionGroupISidRowStatus
}
STATUS current
DESCRIPTION
    "Objects for the ieee8021PbbTeProtectionGroupISidGroup group."
 ::= { ieee8021PbbTeGroups 7 }

ieee8021PbbTeFdbGroup OBJECT-GROUP
OBJECTS {
    ieee8021PbbTeBridgeStaticForwardAnyUnicastEgressPorts,
    ieee8021PbbTeBridgeStaticForwardAnyUnicastForbiddenPorts,
    ieee8021PbbTeBridgeStaticForwardAnyUnicastStorageType,
    ieee8021PbbTeBridgeStaticForwardAnyUnicastRowStatus
}
STATUS current
DESCRIPTION
    "Fdb extension objects group"
 ::= { ieee8021PbbTeGroups 8 }

ieee8021PbbTeNotificationsGroup NOTIFICATION-GROUP
NOTIFICATIONS {
    ieee8021PbbTeProtectionGroupAdminFailure
}
STATUS current
DESCRIPTION
    "Objects for the notifications group."
 ::= { ieee8021PbbTeGroups 9 }

ieee8021PbbTeCompliance MODULE-COMPLIANCE
STATUS current
DESCRIPTION
    "The compliance statement for support of the PBB-TE MIB module."
MODULE
MANDATORY-GROUPS {
    ieee8021PbbTeGroupListGroup,
    ieee8021PbbTeMASharedGroup,
    ieee8021PbbTeTesiGroup,
    ieee8021PbbTeSiEspGroup,
    ieee8021PbbTeProtectionConfigManGroup,
    ieee8021PbbTeProtectionGroupISidGroup,
    ieee8021PbbTeFdbGroup,
    ieee8021PbbTeNotificationsGroup
}

```

```
        }
GROUP ieee8021PbbTeProtectionConfigOptGroup
DESCRIPTION
    "This group allows implementation to optionally change the
     WaitToRestore and HoldOff timers for protection groups."
OBJECT ieee8021PbbTeProtectionGroupConfigWTR
MIN-ACCESS not-accessible
DESCRIPTION "This object is optional."
OBJECT ieee8021PbbTeProtectionGroupConfigHoldOff
MIN-ACCESS not-accessible
DESCRIPTION "This object is optional."
OBJECT ieee8021PbbTeProtectionGroupListRowStatus
SYNTAX      RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                        destroy(6) }
DESCRIPTION "Support for createAndWait is not required."
OBJECT ieee8021PbbTeTeSiEspRowStatus
SYNTAX      RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                        destroy(6) }
DESCRIPTION "Support for createAndWait is not required."
OBJECT ieee8021PbbTeProtectionGroupISidRowStatus
SYNTAX      RowStatus { active(1), notInService(2) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                        destroy(6) }
DESCRIPTION "Support for createAndWait is not required."
 ::= { ieee8021PbbTeCompliances 1 }
```

END

17.7.11 Definitions for the TPMR MIB module

In the MIB definition below, if any discrepancy between the DESCRIPTION text and the corresponding definition in 12.19 occur, the definition in 12.19 takes precedence.

```

IEEE8021-TPMR-MIB DEFINITIONS ::= BEGIN

-- -----
-- MIB for IEEE 802.1aj TPMR Devices
-- -----


IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Counter64,
    Unsigned32
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION, TruthValue, MacAddress, TimeInterval,
    StorageType
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    ifCounterDiscontinuityGroup
        FROM IF-MIB
    IEEE8021BridgePortNumber, ieee802dot1mibs
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBasePortComponentId
        FROM IEEE8021-BRIDGE-MIB;

ieee8021TpmrMib MODULE-IDENTITY
    LAST-UPDATED "201102270000Z" -- February 27, 2011
    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: Brian Hassink
        Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
        E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
    "Two-Port MAC Relay (TPMR) MIB module.

    Unless otherwise indicated, the references in this MIB
    module are to IEEE 802.1Q-2011.

    Copyright (c) IEEE
    This MIB module is part of IEEE Std. 802.1aj; please
    refer to the document itself for full legal notices."
REVISION      "201102270000Z" -- February 27, 2011
DESCRIPTION
    "Minor edits to contact information etc. as part of
    2011 revision of IEEE Std 802.1Q."
REVISION      "200909040000Z" -- September 4, 2009

```

```
DESCRIPTION
    "Initial version as published in IEEE Std. 802.1aj"
 ::= { ieee802dot1mib 14 }

ieee8021TpmlNotifications OBJECT IDENTIFIER ::= { ieee8021TpmlMib 0 }
ieee8021TpmlObjects      OBJECT IDENTIFIER ::= { ieee8021TpmlMib 1 }
ieee8021TpmlConformance  OBJECT IDENTIFIER ::= { ieee8021TpmlMib 2 }

-- -----
-- Textual conventions
-- -----
```

IEEE8021TpmlFrameDiscardErrorReason ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
 "A reason code for a frame discard error."
REFERENCE
 "IEEE 802.1aj 12.19.3.1.1.3:h"
SYNTAX INTEGER {
 txSduSizeExceeded (1) -- transmissible SDU size exceeded
}

```
-- -----
-- ieee8021TpmlPort objects
-- -----
```

ieee8021TpmlPortTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021TpmlPortEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "The TPMR port table. Each row in the table represents a
 TPMR port. By definition there are two ports per TPMR.

 Note that the indices of this table are equivalent to
 those of the ieee8021BridgeBasePortTable in the
 IEEE8021-BRIDGE-MIB, with ieee8021TpmlPortNumber having
 a more limited value range than ieee8021BridgeBasePort."
REFERENCE
 "IEEE 802.1aj 12.19.1.2.1 and 12.19.1.2.2"
 ::= { ieee8021TpmlObjects 1 }

ieee8021TpmlPortEntry OBJECT-TYPE
SYNTAX Ieee8021TpmlPortEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "A TPMR port table entry."
INDEX { ieee8021BridgeBasePortComponentId,
 ieee8021TpmlPortNumber }
 ::= { ieee8021TpmlPortTable 1 }

Ieee8021TpmlPortEntry ::= SEQUENCE {
 ieee8021TpmlPortNumber IEEE8021BridgePortNumber,
 ieee8021TpmlPortMgmtAddr TruthValue,
 ieee8021TpmlPortMgmtAddrForwarding TruthValue
}

ieee8021TpmlPortNumber OBJECT-TYPE

MAC BRIDGES AND VIRTUAL BRIDGED LOCAL AREA NETWORKS

```
SYNTAX      IEEE8021BridgePortNumber (1..2)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The number of this TPMR port."
REFERENCE
    "IEEE 802.1aj 12.19.1.1.3:b,1"
::= { ieee8021TpmpPortEntry 1 }

ieee8021TpmpPortMgmtAddr OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Is 'true' if the TPMR port MAC address is the management
     address of the TPMR, otherwise 'false'."
REFERENCE
    "IEEE 802.1aj 12.19.1.1.3:b,3"
::= { ieee8021TpmpPortEntry 2 }

ieee8021TpmpPortMgmtAddrForwarding OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "Is 'true' if forwarding is enabled for frames destined to the
     management address of the TPMR, otherwise 'false'."
REFERENCE
    "IEEE 802.1aj 12.19.1.2.1.3:c"
::= { ieee8021TpmpPortEntry 3 }

-- -----
-- ieee8021TpmpPortStats objects
-- -----


ieee8021TpmpPortStatsTable OBJECT-TYPE
SYNTAX SEQUENCE OF Ieee8021TpmpPortStatsEntry
MAX-ACCESS      not-accessible
STATUS         current
DESCRIPTION
    "The TPMR port statistics table. Each row in the table
     represents a TPMR port. By definition there are two
     ports per TPMR.

     Discontinuities in the value of counters in this table
     can occur at re-initialization of the management system,
     and at other times as indicated by the value of IF-MIB
     ifCounterDiscontinuityTime."
REFERENCE
    "IEEE 802.1aj 12.19.3.1"
::= { ieee8021TpmpObjects 2 }

ieee8021TpmpPortStatsEntry OBJECT-TYPE
SYNTAX      Ieee8021TpmpPortStatsEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A TPMR port counters table entry."
AUGMENTS { ieee8021TpmpPortEntry }
```

```
 ::= { ieee8021TpmpPortStatsTable 1 }

Ieee8021TpmpPortStatsEntry ::= SEQUENCE {
    ieee8021TpmpPortStatsRxFrames           Counter64,
    ieee8021TpmpPortStatsRxOctets           Counter64,
    ieee8021TpmpPortStatsFramesForwarded   Counter64,
    ieee8021TpmpPortStatsFramesDiscarded   Counter64,
    ieee8021TpmpPortStatsFramesDiscardedQueueFull Counter64,
    ieee8021TpmpPortStatsFramesDiscardedLifetime Counter64,
    ieee8021TpmpPortStatsFramesDiscardedError   Counter64
}

ieee8021TpmpPortStatsRxFrames OBJECT-TYPE
    SYNTAX      Counter64
    UNITS       "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Count of all valid frames received on this port (including
         BPDUs, frames addressed to the TPMR as an end station, and
         frames that were submitted to the Forwarding Process)."
    REFERENCE
        "IEEE 802.1aj 12.19.3.1.1.3:a"
    ::= { ieee8021TpmpPortStatsEntry 1 }

ieee8021TpmpPortStatsRxOctets OBJECT-TYPE
    SYNTAX      Counter64
    UNITS       "octets"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Count of the total number of octets in all valid frames
         received on this port (including BPDUs, frames addressed
         to the TPMR as an end station, and frames that were
         submitted to the Forwarding Process)."
    REFERENCE
        "IEEE 802.1aj 12.19.3.1.1.3:b"
    ::= { ieee8021TpmpPortStatsEntry 2 }

ieee8021TpmpPortStatsFramesForwarded OBJECT-TYPE
    SYNTAX      Counter64
    UNITS       "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Count of all frames that were received on this port and
         were forwarded to the transmission port."
    REFERENCE
        "IEEE 802.1aj 12.19.3.1.1.3:d"
    ::= { ieee8021TpmpPortStatsEntry 3 }

ieee8021TpmpPortStatsFramesDiscarded OBJECT-TYPE
    SYNTAX      Counter64
    UNITS       "frames"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "Count of all frames that were received on this port but
         were discarded by the Forwarding Process for any reason."
```

```

REFERENCE
  "IEEE 802.1aj 12.19.3.1.1.3:c"
 ::= { ieee8021TpmpPortStatsEntry 4 }

ieee8021TpmpPortStatsFramesDiscardedQueueFull OBJECT-TYPE
  SYNTAX      Counter64
  UNITS      "frames"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Count of all frames received on this port that were to
     be transmitted through the transmission port but were
     discarded due to lack of available queue space."
REFERENCE
  "IEEE 802.1aj 12.19.3.1.1.3:e"
 ::= { ieee8021TpmpPortStatsEntry 5 }

ieee8021TpmpPortStatsFramesDiscardedLifetime OBJECT-TYPE
  SYNTAX      Counter64
  UNITS      "frames"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Count of all frames received on this port that were to
     be transmitted through the transmission port but were
     discarded due to their frame lifetime having been
     exceeded."
REFERENCE
  "IEEE 802.1aj 12.19.3.1.1.3:f"
 ::= { ieee8021TpmpPortStatsEntry 6 }

ieee8021TpmpPortStatsFramesDiscardedError OBJECT-TYPE
  SYNTAX      Counter64
  UNITS      "frames"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "Count of all frames received on this port that were to
     be transmitted through the transmission port but could
     not be transmitted (e.g., frame too large)."
REFERENCE
  "IEEE 802.1aj 12.19.3.1.1.3:g"
 ::= { ieee8021TpmpPortStatsEntry 7 }

-- -----
-- ieee8021TpmpPortDiscardDetails objects
-- -----
```

ieee8021TpmpPortDiscardDetailsTable OBJECT-TYPE

SYNTAX SEQUENCE OF Ieee8021TpmpPortDiscardDetailsEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"The TPMR frames discard details table. Each row in the table represents a discarded frame on a TPMR port. By definition there are two ports per TPMR.

This table is maintained as a FIFO. A new entry is inserted in the first row, and existing entries are

shuffled down, with the last entry being discarded.

Because of the FIFO behaviour, the relationship between the index and contents will change when an entry is added to the table. This may result in apparent duplication of row content during a table traversal."

REFERENCE

"IEEE 802.1aj 12.19.3.1.1.3:h"
 ::= { ieee8021TpmpmrObjects 3 }

ieee8021TpmpmrPortDiscardDetailsEntry OBJECT-TYPE
 SYNTAX Ieee8021TpmpmrPortDiscardDetailsEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "A TPMR frames discarded error details table entry."
 INDEX { ieee8021BridgeBasePortComponentId,
 ieee8021TpmpmrPortNumber,
 ieee8021TpmpmrPortDiscardDetailsIndex }
 ::= { ieee8021TpmpmrPortDiscardDetailsTable 1 }

Ieee8021TpmpmrPortDiscardDetailsEntry ::= SEQUENCE {
 ieee8021TpmpmrPortDiscardDetailsIndex Unsigned32,
 ieee8021TpmpmrPortDiscardDetailsSource MacAddress,
 ieee8021TpmpmrPortDiscardDetailsReason IEEE8021TpmpmrFrameDiscardErrorReason
 }

ieee8021TpmpmrPortDiscardDetailsIndex OBJECT-TYPE
 SYNTAX Unsigned32 (1..16)
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "The second index of a TPMR frames discard details table entry."
 ::= { ieee8021TpmpmrPortDiscardDetailsEntry 1 }

ieee8021TpmpmrPortDiscardDetailsSource OBJECT-TYPE
 SYNTAX MacAddress
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The source MAC address of the discarded frame."
 REFERENCE
 "IEEE 802.1aj 12.19.3.1.1.3:h"
 ::= { ieee8021TpmpmrPortDiscardDetailsEntry 2 }

ieee8021TpmpmrPortDiscardDetailsReason OBJECT-TYPE
 SYNTAX IEEE8021TpmpmrFrameDiscardErrorReason
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The reason why the frame was discarded."
 REFERENCE
 "IEEE 802.1aj 12.19.3.1.1.3:h"
 ::= { ieee8021TpmpmrPortDiscardDetailsEntry 3 }

-- -----
-- ieee8021TpmpmrMsp objects
-- -----

```

ieee8021TpmpRspTable OBJECT-TYPE
  SYNTAX  SEQUENCE OF Ieee8021TpmpRspEntry
  MAX-ACCESS    not-accessible
  STATUS       current
  DESCRIPTION
    "The TPMR MAC status propagation performance table. Each
     row in the table represents a TPMR port. By definition
     there are two ports per TPMR."

```

The persistence of writable objects in a conceptual row of this table is determined by the value of the ieee8021TpmpRspStorageType object."

REFERENCE
 "IEEE 802.1aj 12.19.4.1.1 and 12.19.4.1.2"
 ::= { ieee8021TpmpObjects 4 }

```

ieee8021TpmpRspEntry OBJECT-TYPE
  SYNTAX      Ieee8021TpmpRspEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "A TPMR MAC status propagation performance table entry."
  AUGMENTS { ieee8021TpmpPortEntry }
  ::= { ieee8021TpmpRspTable 1 }

```

```

Ieee8021TpmpRspEntry ::= SEQUENCE {
  ieee8021TpmpRspLinkNotify      TruthValue,
  ieee8021TpmpRspLinkNotifyWait  TimeInterval,
  ieee8021TpmpRspLinkNotifyRetry TimeInterval,
  ieee8021TpmpRspMacNotify      TruthValue,
  ieee8021TpmpRspMacNotifyTime   TimeInterval,
  ieee8021TpmpRspMacRecoverTime TimeInterval,
  ieee8021TpmpRspStorageType     StorageType
}

```

```

ieee8021TpmpRspLinkNotify OBJECT-TYPE
  SYNTAX      TruthValue
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "The value of LinkNotify used by the MSP state machines."
  REFERENCE
    "IEEE 802.1aj 12.19.4.1.1.3:a and 12.19.4.1.2.2:b"
  DEFVAL { true }
  ::= { ieee8021TpmpRspEntry 1 }

```

```

ieee8021TpmpRspLinkNotifyWait OBJECT-TYPE
  SYNTAX      TimeInterval (20..100)
  UNITS      "centiseconds"
  MAX-ACCESS  read-write
  STATUS      current
  DESCRIPTION
    "The value of LinkNotifyWait used by the MSP state machines."
  REFERENCE
    "IEEE 802.1aj 12.19.4.1.1.3:b and 12.19.4.1.2.2:c"
  DEFVAL { 40 }
  ::= { ieee8021TpmpRspEntry 2 }

```

```
ieee8021TpMrMspLinkNotifyRetry OBJECT-TYPE
    SYNTAX      TimeInterval (10..100)
    UNITS      "centiseconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of LinkNotifyRetry used by the MSP state machines."
    REFERENCE
        "IEEE 802.1aj 12.19.4.1.1.3:c and 12.19.4.1.2.2:d"
    DEFVAL { 100 }
    ::= { ieee8021TpMrMspEntry 3 }

ieee8021TpMrMspMacNotify OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of MacNotify used by the MSP state machines."
    REFERENCE
        "IEEE 802.1aj 12.19.4.1.1.3:d and 12.19.4.1.2.2:e"
    DEFVAL { true }
    ::= { ieee8021TpMrMspEntry 4 }

ieee8021TpMrMspMacNotifyTime OBJECT-TYPE
    SYNTAX      TimeInterval (1..50)
    UNITS      "centiseconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of MacNotifyTime used by the MSP state machines."
    REFERENCE
        "IEEE 802.1aj 12.19.4.1.1.3:e and 12.19.4.1.2.2:f"
    DEFVAL { 20 }
    ::= { ieee8021TpMrMspEntry 5 }

ieee8021TpMrMspMacRecoverTime OBJECT-TYPE
    SYNTAX      TimeInterval (2..50)
    UNITS      "centiseconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of MacRecoverTime used by the MSP state machines."
    REFERENCE
        "IEEE 802.1aj 12.19.4.1.1.3:f and 12.19.4.1.2.2:g"
    DEFVAL { 10 }
    ::= { ieee8021TpMrMspEntry 6 }

ieee8021TpMrMspStorageType OBJECT-TYPE
    SYNTAX      StorageType
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The storage type for all read-write objects within this
         row. Conceptual rows having the value 'permanent' need
         not allow write access to any columnar objects in the row.

        If this object has the value 'volatile', modifications
         to read-write objects in this row are not persistent
         across reboots or restarts. If this object has the value
```

```

    'nonVolatile', modifications to objects in this row
    are persistent."
DEFVAL { nonVolatile }
 ::= { ieee8021TpmpRMspEntry 7 }

-- -----
-- ieee8021TpmpRMspStats objects
-- -----


ieee8021TpmpRMspStatsTable OBJECT-TYPE
    SYNTAX SEQUENCE OF Ieee8021TpmpRMspStatsEntry
    MAX-ACCESS      not-accessible
    STATUS         current
    DESCRIPTION
        "The TPMR MAC status propagation statistics table. Each
        row in the table represents a TPMR port. By definition
        there are two ports per TPMR.

        Discontinuities in the value of counters in this table
        can occur at re-initialization of the management system,
        and at other times as indicated by the value of IF-MIB
        ifCounterDiscontinuityTime."
    REFERENCE
        "IEEE 802.1aj 12.19.4.1.3"
 ::= { ieee8021TpmpRObjects 5 }

ieee8021TpmpRMspStatsEntry OBJECT-TYPE
    SYNTAX      Ieee8021TpmpRMspStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A TPMR MAC status propagation statistics table entry."
    AUGMENTS { ieee8021TpmpRPortEntry }
 ::= { ieee8021TpmpRMspStatsTable 1 }

Ieee8021TpmpRMspStatsEntry ::= SEQUENCE {
    ieee8021TpmpRMspStatsTxAcks          Counter32,
    ieee8021TpmpRMspStatsTxAddNotifications Counter32,
    ieee8021TpmpRMspStatsTxAddConfirmations Counter32,
    ieee8021TpmpRMspStatsTxLossNotifications Counter32,
    ieee8021TpmpRMspStatsTxLossConfirmations Counter32,
    ieee8021TpmpRMspStatsRxAcks          Counter32,
    ieee8021TpmpRMspStatsRxAddNotifications Counter32,
    ieee8021TpmpRMspStatsRxAddConfirmations Counter32,
    ieee8021TpmpRMspStatsRxLossNotifications Counter32,
    ieee8021TpmpRMspStatsRxLossConfirmations Counter32,
    ieee8021TpmpRMspStatsAddEvents        Counter32,
    ieee8021TpmpRMspStatsLossEvents       Counter32,
    ieee8021TpmpRMspStatsMacStatusNotifications Counter32
}

ieee8021TpmpRMspStatsTxAcks OBJECT-TYPE
    SYNTAX      Counter32
    UNITS      "MSPDUs"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of acks transmitted by the port's Transmit
        Process as a consequence of txAck being set."

```

```
REFERENCE
    "IEEE 802.1aj 12.19.4.1.3.3:a"
 ::= { ieee8021TpmpRStatsEntry 1 }

ieee8021TpmpRStatsTxAddNotifications OBJECT-TYPE
    SYNTAX      Counter32
    UNITS      "MSPDUs"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of adds transmitted by the port's Transmit
         Process as a consequence of txAdd being set."
REFERENCE
    "IEEE 802.1aj 12.19.4.1.3.3:b"
 ::= { ieee8021TpmpRStatsEntry 2 }

ieee8021TpmpRStatsTxAddConfirmations OBJECT-TYPE
    SYNTAX      Counter32
    UNITS      "MSPDUs"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of add confirms transmitted by the port's
         Transmit Process as a consequence of txAddConfirm
         being set."
REFERENCE
    "IEEE 802.1aj 12.19.4.1.3.3:c"
 ::= { ieee8021TpmpRStatsEntry 3 }

ieee8021TpmpRStatsTxLossNotifications OBJECT-TYPE
    SYNTAX      Counter32
    UNITS      "MSPDUs"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of losses transmitted by the port's Transmit
         Process as a consequence of txLoss being set."
REFERENCE
    "IEEE 802.1aj 12.19.4.1.3.3:d"
 ::= { ieee8021TpmpRStatsEntry 4 }

ieee8021TpmpRStatsTxLossConfirmations OBJECT-TYPE
    SYNTAX      Counter32
    UNITS      "MSPDUs"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The number of loss confirms transmitted by the port's
         Transmit Process as a consequence of txLossConfirm
         being set."
REFERENCE
    "IEEE 802.1aj 12.19.4.1.3.3:e"
 ::= { ieee8021TpmpRStatsEntry 5 }

ieee8021TpmpRStatsRxAcks OBJECT-TYPE
    SYNTAX      Counter32
    UNITS      "MSPDUs"
    MAX-ACCESS  read-only
    STATUS      current
```

```

DESCRIPTION
  "The number of acks received by the port's Receive
  Process."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:f"
 ::= { ieee8021TpmrMspStatsEntry 6 }

ieee8021TpmrMspStatsRxAddNotifications OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSPDUs"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The number of adds received by the port's Receive
     Process."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:g"
 ::= { ieee8021TpmrMspStatsEntry 7 }

ieee8021TpmrMspStatsRxAddConfirmations OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSPDUs"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The number of add confirms received by the port's
     Receive Process."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:h"
 ::= { ieee8021TpmrMspStatsEntry 8 }

ieee8021TpmrMspStatsRxLossNotifications OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSPDUs"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The number of losses received by the port's Receive
     Process."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:i"
 ::= { ieee8021TpmrMspStatsEntry 9 }

ieee8021TpmrMspStatsRxLossConfirmations OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSPDUs"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The number of loss confirms received by the port's
     Receive Process."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:j"
 ::= { ieee8021TpmrMspStatsEntry 10 }

ieee8021TpmrMspStatsAddEvents OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSP transitions"
  MAX-ACCESS  read-only

```

```
STATUS      current
DESCRIPTION
  "The number of transitions to STM:ADD directly from
   STM:DOWN or STM:LOSS."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:k"
 ::= { ieee8021TpmrMspStatsEntry 11 }
```

```
ieee8021TpmrMspStatsLossEvents OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSP transitions"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The number of transitions to STM:LOSS directly from
     STM:UP or STM:ADD."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:l"
 ::= { ieee8021TpmrMspStatsEntry 12 }
```

```
ieee8021TpmrMspStatsMacStatusNotifications OBJECT-TYPE
  SYNTAX      Counter32
  UNITS      "MSP transitions"
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The number of transitions to SNM:MAC_NOTIFICATION."
REFERENCE
  "IEEE 802.1aj 12.19.4.1.3.3:m"
 ::= { ieee8021TpmrMspStatsEntry 13 }
```

```
-- -----
-- IEEE 802.1aj MIB - Conformance Information
-- -----
```

```
ieee8021TpmrCompliances OBJECT IDENTIFIER ::= { ieee8021TpmrConformance 1 }
ieee8021TpmrGroups      OBJECT IDENTIFIER ::= { ieee8021TpmrConformance 2 }
```

```
-- -----
-- Units of conformance
-- -----
```

```
ieee8021TpmrPortGroup OBJECT-GROUP
  OBJECTS {
    ieee8021TpmrPortMgmtAddr,
    ieee8021TpmrPortMgmtAddrForwarding
  }
  STATUS current
  DESCRIPTION
    "TPMR port objects."
 ::= { ieee8021TpmrGroups 1 }
```

```
ieee8021TpmrPortStatsGroup OBJECT-GROUP
  OBJECTS {
    ieee8021TpmrPortStatsRxFrames,
    ieee8021TpmrPortStatsRxOctets,
    ieee8021TpmrPortStatsFramesForwarded,
    ieee8021TpmrPortStatsFramesDiscarded,
    ieee8021TpmrPortStatsFramesDiscardedQueueFull,
```

```

ieee8021TpmpPortStatsFramesDiscardedLifetime,
ieee8021TpmpPortStatsFramesDiscardedError
}
STATUS current
DESCRIPTION
    "TPMR port statistics objects."
::= { ieee8021TpmpGroups 2 }

ieee8021TpmpPortDiscardDetailsGroup OBJECT-GROUP
OBJECTS {
    ieee8021TpmpPortDiscardDetailsSource,
    ieee8021TpmpPortDiscardDetailsReason
}
STATUS current
DESCRIPTION
    "TPMR port discard details objects."
::= { ieee8021TpmpGroups 3 }

ieee8021TpmpMspGroup OBJECT-GROUP
OBJECTS {
    ieee8021TpmpMspLinkNotify,
    ieee8021TpmpMspLinkNotifyWait,
    ieee8021TpmpMspLinkNotifyRetry,
    ieee8021TpmpMspMacNotify,
    ieee8021TpmpMspMacNotifyTime,
    ieee8021TpmpMspMacRecoverTime,
    ieee8021TpmpMspStorageType
}
STATUS current
DESCRIPTION
    "TPMR port MSP objects."
::= { ieee8021TpmpGroups 4 }

ieee8021TpmpMspStatsGroup OBJECT-GROUP
OBJECTS {
    ieee8021TpmpMspStatsTxAcks,
    ieee8021TpmpMspStatsTxAddNotifications,
    ieee8021TpmpMspStatsTxAddConfirmations,
    ieee8021TpmpMspStatsTxLossNotifications,
    ieee8021TpmpMspStatsTxLossConfirmations,
    ieee8021TpmpMspStatsRxAcks,
    ieee8021TpmpMspStatsRxAddNotifications,
    ieee8021TpmpMspStatsRxAddConfirmations,
    ieee8021TpmpMspStatsRxLossNotifications,
    ieee8021TpmpMspStatsRxLossConfirmations,
    ieee8021TpmpMspStatsAddEvents,
    ieee8021TpmpMspStatsLossEvents,
    ieee8021TpmpMspStatsMacStatusNotifications
}
STATUS current
DESCRIPTION
    "TPMR port MSP statistics objects."
::= { ieee8021TpmpGroups 5 }

-- -----
-- Compliance statements
-- -----

```

ieee8021TpmpCompliance MODULE-COMPLIANCE

```
STATUS current
DESCRIPTION
    "The compliance statement for device support of TPMR."
MODULE IF-MIB
    MANDATORY-GROUPS {
        ifCounterDiscontinuityGroup
    }
MODULE
    MANDATORY-GROUPS {
        ieee8021TpmrPortGroup,
        ieee8021TpmrPortStatsGroup,
        ieee8021TpmrPortDiscardDetailsGroup,
        ieee8021TpmrMspGroup,
        ieee8021TpmrMspStatsGroup
    }
 ::= { ieee8021TpmrCompliances 1 }

END
```

17.7.12 Definitions for the IEEE8021-FQTSS MIB module

```

IEEE8021-FQTSS-MIB DEFINITIONS ::= BEGIN

-- =====
-- MIB for support of 802.1Qav Forwarding & Queuing Enhancements
-- for Time Sensitive Streams (FQTSS) in 802.1Q Bridges.
-- =====

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION,
    TruthValue,
    RowStatus
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    ieee802dot1mibs,
    IEEE8021PriorityValue
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBaseComponentId,
    ieee8021BridgeBasePort
        FROM IEEE8021-BRIDGE-MIB
;

ieee8021FqtssMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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: Tony Jeffree
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
    "The Bridge MIB module for managing devices that support
    the Forwarding and Queuing Enhancements
    for Time Sensitive Streams.

Unless otherwise indicated, the references in this MIB
module are to IEEE Std 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
REVISION      "201102270000Z" -- February 27, 2011

```

DESCRIPTION

"Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."

REVISION "200910010000Z" -- October 1, 2009

DESCRIPTION

"Initial revision, included in IEEE 802.1Qav."
 ::= { ieee802dot1mibs 16 }

-- =====
-- Textual Conventions
-- =====

IEEE8021FqtssTrafficClassValue ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"
STATUS current
DESCRIPTION

"An 802.1 FQTSS traffic class value.
This is the numerical value associated with a traffic
class in a Bridge. Larger values are associated with
higher priority traffic classes."
REFERENCE "12.21"
SYNTAX Unsigned32 (0..7)

IEEE8021FqtssDeltaBandwidthValue ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"
STATUS current
DESCRIPTION

"An 802.1 FQTSS delta bandwidth percentage,
represented as a fixed point number scaled by
1,000,000."
REFERENCE "12.21, 34.4"
SYNTAX Unsigned32 (0..100000000)

IEEE8021FtqssTxSelectionAlgorithmIDValue ::= TEXTUAL-CONVENTION

DISPLAY-HINT "d"
STATUS current
DESCRIPTION

"An 802.1 transmission selection algorithm identifier
value. This is an integer, with the following
interpretation placed on the value:

0: Strict priority algorithm,
1: Credit-based shaper algorithm,
2-255: Reserved for future standardization,
256-4294967295: Vendor-specific transmission selection
algorithm identifiers, consisting of a
four-octet integer, where the 3 most
significant octets hold an OUI value,
and the least significant octet holds
an integer value in the range 0-255
assigned by the owner of the OUI."
REFERENCE "8.6.8, 12.21"
SYNTAX Unsigned32

-- =====
-- subtrees in the FQTSS MIB
-- =====

```

ieee8021FqtssNotifications
    OBJECT IDENTIFIER ::= { ieee8021FqtssMib 0 }

ieee8021FqtssObjects
    OBJECT IDENTIFIER ::= { ieee8021FqtssMib 1 }

ieee8021FqtssConformance
    OBJECT IDENTIFIER ::= { ieee8021FqtssMib 2 }

ieee8021FqtssBap
    OBJECT IDENTIFIER ::= { ieee8021FqtssObjects 1 }

ieee8021FqtssMappings
    OBJECT IDENTIFIER ::= { ieee8021FqtssObjects 2 }

-- =====
-- The ieee8021FqtssBap subtree
-- This subtree defines the objects necessary for the management
-- of bandwidth allocation for queues that support FQTSS.
-- =====

-- =====
-- the ieee8021FqtssBapTable
-- =====

ieee8021FqtssBapTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssBapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of bandwidth availability
         parameters for each traffic class that supports the
         credit-based shaper algorithm.
         All writable objects in this table must be
         persistent over power up restart/reboot."
    REFERENCE   "12.21.1"
    ::= { ieee8021FqtssBap 1 }

ieee8021FqtssBapEntry OBJECT-TYPE
    SYNTAX      Ieee8021FqtssBapEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing bandwidth allocation
         information for each traffic class that supports the
         credit-based shaper algorithm. Rows in the table are
         automatically created and deleted as a result of the
         operation of the algorithm described in 34.5. "
    INDEX    { ieee8021BridgeBaseComponentId,
              ieee8021BridgeBasePort,
              ieee8021FqtssBAPTrafficClass }
    ::= { ieee8021FqtssBapTable 1 }

Ieee8021FqtssBapEntry :=
    SEQUENCE {
        ieee8021FqtssBAPTrafficClass
            IEEE8021FqtssTrafficClassValue,
        ieee8021FqtssDeltaBandwidth
            IEEE8021FqtssDeltaBandwidthValue,
    }

```

```
ieee8021FqtssOperIdleSlopeMs
    Unsigned32,
ieee8021FqtssOperIdleSlopeLs
    Unsigned32,
ieee8021FqtssAdminIdleSlopeMs
    Unsigned32,
ieee8021FqtssAdminIdleSlopeLs
    Unsigned32,
ieee8021FqtssBapRowStatus
    RowStatus
}

ieee8021FqtssBAPTrafficClass OBJECT-TYPE
SYNTAX      IEEE8021FqtssTrafficClassValue
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The traffic class number associated with the row of
the table.

A row in this table is created for each traffic class
that supports the credit-based shaper algorithm. The
recommended mappings of priorities to traffic classes
for support of the credit-based shaper algorithm are
described in 34.5."
REFERENCE    "12.21.1, 34.3, 34.5"
 ::= { ieee8021FqtssBapEntry 1 }
```

```
ieee8021FqtssDeltaBandwidth OBJECT-TYPE
SYNTAX      IEEE8021FqtssDeltaBandwidthValue
UNITS      "percent"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The value of the deltaBandwidth parameter
for the traffic class.
This value is represented as a fixed point number
scaled by a factor of 1,000,000; i.e., 100,000,000
(the maximum value) represents 100%.
```

The default value of the deltaBandwidth parameter for the highest numbered traffic class that supports the credit-based shaper algorithm is 75%; for all lower numbered traffic classes that support the credit-based shaper algorithm the default value is 0%.

The value of this object MUST be retained across reinitializations of the management system."

```
REFERENCE    "12.21.1, 34.3"
 ::= { ieee8021FqtssBapEntry 2}
```

```
ieee8021FqtssOperIdleSlopeMs OBJECT-TYPE
SYNTAX      Unsigned32
UNITS      "bits per second"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The most significant 32 bits of the bandwidth,
```

in bits per second, that is currently allocated to the traffic class (idleSlope(N)). This object MUST be read at the same time as ieee8021FqtssOperIdleSlopeLs, which represents the LS 32 bits of the value, in order for the read operation to succeed.

If SRP is supported and in operation, then the reserved bandwidth is determined by the operation of SRP; otherwise, the value of ieee8021FqtssOperIdleSlopeMs is equal to the value of ieee8021FqtssAdminIdleSlopeMs.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.21.1, 34.3"

`::= { ieee8021FqtssBapEntry 3 }`

`ieee8021FqtssOperIdleSlopeLs` OBJECT-TYPE

SYNTAX Unsigned32

UNITS "bits per second"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The least significant 32 bits of the bandwidth, in bits per second, that is currently allocated to the traffic class (idleSlope(N)). This object MUST be read at the same time as ieee8021FqtssOperIdleSlopeMs, which represents the LS 32 bits of the value, in order for the read operation to succeed.

If SRP is supported and in operation, then the reserved bandwidth is determined by the operation of SRP; otherwise, the value of ieee8021FqtssOperIdleSlopeLs is equal to the value of ieee8021FqtssAdminIdleSlopeMs.

The value of this object MUST be retained across reinitializations of the management system."

REFERENCE "12.21.1, 34.3"

`::= { ieee8021FqtssBapEntry 4 }`

`ieee8021FqtssAdminIdleSlopeMs` OBJECT-TYPE

SYNTAX Unsigned32

UNITS "bits per second"

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"The most significant 32 bits of the bandwidth, in bits per second, that the manager desires to allocate to the traffic class as idleSlope(N). This object MUST be read or written at the same time as ieee8021FqtssAdminIdleSlopeLs, which represents the LS 32 bits of the value, in order for the read or write operation to succeed.

If SRP is supported and in operation, then the reserved bandwidth is determined by the operation of SRP, and any changes to the value of this object have no effect on the operational value of idleSlope(N).

The value of this object MUST be retained across

```
    reinitializations of the management system."
REFERENCE    "12.21.1, 34.3"
DEFVAL { 0 }
 ::= { ieee8021FqtssBapEntry 5 }

ieee8021FqtssAdminIdleSlopeLs OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "bits per second"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "The least significant 32 bits of the bandwidth,
     in bits per second, that the manager desires to allocate
     to the traffic class as idleSlope(N). This object MUST be
     read or written at the same time as
     ieee8021FqtssAdminIdleSlopeMs,
     which represents the LS 32 bits of the value, in order
     for the read or write operation to succeed.

If SRP is supported and in operation, then the reserved
bandwidth is determined by the operation of SRP, and any
changes to the value of this object have no effect on the
operational value of idleSlope(N).

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE    "12.21.1, 34.3"
DEFVAL { 0 }
 ::= { ieee8021FqtssBapEntry 6 }

ieee8021FqtssBapRowStatus OBJECT-TYPE
SYNTAX RowStatus
MAX-ACCESS read-create
STATUS current
DESCRIPTION
    "Indicates the status of an entry (row) in this table, and is
     used to create/delete entries.

The corresponding instances of the following objects
must be set before this object can be made active(1):
    ieee8021FqtssBAPTrafficClass
    ieee8021FqtssDeltaBandwidth
    ieee8021FqtssOperIdleSlopeMs
    ieee8021FqtssOperIdleSlopeLs
    ieee8021FqtssAdminIdleSlopeMs
    ieee8021FqtssAdminIdleSlopeLs

The corresponding instances of the following objects
may not be changed while this object is active(1):
    ieee8021FqtssBAPTrafficClass"
 ::= { ieee8021FqtssBapEntry 7 }

-- =====
-- The ieee8021FqtssMappings subtree
-- This subtree defines the objects necessary for the assignment
-- of transmission selection algorithms to traffic classes,
-- and definition of regeneration table override values.
-- =====
```

```
-- =====
-- the ieee8021FqtssTxSelectionAlgorithmTable
-- =====

ieee8021FqtssTxSelectionAlgorithmTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssTxSelectionAlgorithmEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing the assignment of transmission
         selection algorithms to traffic classes for the Port.
         This table provides management of the Transmission
         Selection Algorithm Table defined in 8.6.8.

For a given Port, a row in the table exists for each
traffic class that is supported by the Port.

The default assignments of transmission selection
algorithms to traffic classes in the table are made
on instantiation of the table, in accordance
with the defaults defined in 8.6.8 and 34.5.

All writable objects in this table must be
persistent over power up restart/reboot."
REFERENCE  "8.6.8, 12.21.2, 34.5"
 ::= { ieee8021FqtssMappings 1 }

ieee8021FqtssTxSelectionAlgorithmEntry OBJECT-TYPE
    SYNTAX      Ieee8021FqtssTxSelectionAlgorithmEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects that contain the mapping of a
         traffic class value to a transmission selection algorithm
         value."
    INDEX   { ieee8021BridgeBaseComponentId,
              ieee8021BridgeBasePort,
              ieee8021FqtssTrafficClass  }
 ::= { ieee8021FqtssTxSelectionAlgorithmTable 1 }

Ieee8021FqtssTxSelectionAlgorithmEntry ::=
SEQUENCE {
    ieee8021FqtssTrafficClass
    IEEE8021FqtssTrafficClassValue,
    ieee8021FqtssTxSelectionAlgorithmID
    IEEE8021FtqssTxSelectionAlgorithmIDValue
}

ieee8021FqtssTrafficClass OBJECT-TYPE
    SYNTAX      IEEE8021FqtssTrafficClassValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The traffic class to which the transmission selection
         algorithm is assigned.

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE  "8.6.8, 12.21.2, 34.5"
```

```
 ::= { ieee8021FqtssTxSelectionAlgorithmEntry 1 }

ieee8021FqtssTxSelectionAlgorithmID OBJECT-TYPE
    SYNTAX      IEEE8021FqtssTxSelectionAlgorithmIDValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The identifier of the transmission selection algorithm
         assigned to the traffic class.

        The value of this object MUST be retained across
         reinitializations of the management system."
    REFERENCE   "8.6.8, 12.21.2, 34.5"
    ::= { ieee8021FqtssTxSelectionAlgorithmEntry 2 }

-- =====
-- the ieee8021FqtssSrpRegenOverrideTable
-- =====

ieee8021FqtssSrpRegenOverrideTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021FqtssSrpRegenOverrideEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing the set of priority regeneration
         table override values for the Port.

        The recommended default values of priorities
         associated with SR classes, and the corresponding
         override values, are defined in 6.9.4.

        All writable objects in this table must be
         persistent over power up restart/reboot."
    REFERENCE   "12.21.3, 6.6.4, 6.9.4"
    ::= { ieee8021FqtssMappings 2 }

Ieee8021FqtssSrpRegenOverrideEntry OBJECT-TYPE
    SYNTAX      Ieee8021FqtssSrpRegenOverrideEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects that contain the mapping of a
         priority value to a priority regeneration override
         value, and a boundary port indication.
         Rows in the table exist for all priorities that are
         associated with SR classes."
    INDEX    { ieee8021BridgeBaseComponentId,
              ieee8021BridgeBasePort,
              ieee8021FqtssSrClassPriority }
    ::= { ieee8021FqtssSrpRegenOverrideTable 1 }

Ieee8021FqtssSrpRegenOverrideEntry :=
    SEQUENCE {
        ieee8021FqtssSrClassPriority
            IEEE8021PriorityValue,
        ieee8021FqtssPriorityRegenOverride
            IEEE8021PriorityValue,
        ieee8021FqtssSrpBoundaryPort
```

```

        TruthValue
    }

ieee8021FqtssSrClassPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The priority value that is overridden at the
         SRP domain boundary. "
    REFERENCE   "12.21.3, 6.6.4, 6.9.4"
    ::= { ieee8021FqtssSrpRegenOverrideEntry 1 }

ieee8021FqtssPriorityRegenOverride OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The priority value that is used to override the
         priority regeneration table entry at the SRP
         domain boundary.

        The value of this object MUST be retained across
         reinitializations of the management system."
    REFERENCE   "12.21.3, 6.6.4, 6.9.4"
    ::= { ieee8021FqtssSrpRegenOverrideEntry 2 }

ieee8021FqtssSrpBoundaryPort OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of the SRPdomainBoundaryPort parameter
         (6.6.4) for the priority. "
    REFERENCE   "12.21.3, 6.6.4, 6.9.4"
    ::= { ieee8021FqtssSrpRegenOverrideEntry 3 }

-- =====
-- IEEE8021 FQTSS MIB - Conformance Information
-- =====

ieee8021FqtssCompliances
    OBJECT IDENTIFIER ::= { ieee8021FqtssConformance 1 }
ieee8021FqtssGroups
    OBJECT IDENTIFIER ::= { ieee8021FqtssConformance 2 }

-- =====
-- units of conformance
-- =====

-- =====
-- the ieee8021FqtssBap group
-- =====

ieee8021FqtssBapGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssDeltaBandwidth,
        ieee8021FqtssOperIdleSlopeMs,
    }

```

```
        ieee8021FqtssOperIdleSlopeLs,
        ieee8021FqtssAdminIdleSlopeMs,
        ieee8021FqtssAdminIdleSlopeLs,
        ieee8021FqtssBapRowStatus
    }
    STATUS      current
    DESCRIPTION
        "Objects that define bandwidth allocation for FQTSS."
    ::= { ieee8021FqtssGroups 1 }

-- =====
-- the ieee8021FqtssTxSelectionAlgorithm group
-- =====

ieee8021FqtssTxSelectionAlgorithmGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssTxSelectionAlgorithmID
    }
    STATUS      current
    DESCRIPTION
        "Objects that define transmission selection
         mappings for FQTSS."
    ::= { ieee8021FqtssGroups 2 }

-- =====
-- the ieee8021FqtssBoundaryPort group
-- =====

ieee8021FqtssBoundaryPortGroup OBJECT-GROUP
    OBJECTS {
        ieee8021FqtssPriorityRegenOverride,
        ieee8021FqtssSrpBoundaryPort
    }
    STATUS      current
    DESCRIPTION
        "Objects that define boundary port priority override
         mappings for FQTSS."
    ::= { ieee8021FqtssGroups 3 }

-- =====
-- compliance statements
-- =====

ieee8021FqtssCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting
         forwarding and queuing for time sensitive streams.

        Support of the objects defined in the IEEE8021-FQTSS MIB
        also requires support of the IEEE8021-BRIDGE-MIB; the
        provisions of 17.3.2 apply to implementations claiming
        support of the IEEE8021-FQTSS MIB. "

    MODULE -- this module
    MANDATORY-GROUPS {
        ieee8021FqtssBapGroup,
        ieee8021FqtssTxSelectionAlgorithmGroup,
        ieee8021FqtssBoundaryPortGroup
    }
```

```
}
```

```
::= { ieee8021FqtssCompliances 1 }
```

```
END
```

17.7.13 Congestion Notification MIB module

In the MIB definition that follows, if any discrepancy between the DESCRIPTION text and the corresponding definition in Clause 12 occur, the definition in Clause 12 takes precedence.

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

-- *****
-- IEEE Std 802.1Qau(TM) Congestion Management MIB
-- *****

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Counter32,
    Counter64,
    Unsigned32          FROM SNMPv2-SMI      -- [RFC2578]
    TEXTUAL-CONVENTION,
    TimeInterval,
    RowStatus,
    TruthValue,
    MacAddress          FROM SNMPv2-TC      -- [RFC2579]
    MODULE-COMPLIANCE,
    OBJECT-GROUP         FROM SNMPv2-CONF     -- [RFC2580]
    InterfaceIndex,
    ifGeneralInformationGroup
                      FROM IF-MIB           -- [RFC2863]
    IEEE8021PriorityValue,
    IEEE8021PbbComponentIdentifier
                      FROM IEEE8021-TC-MIB   -- [IEEE 802.1ap]
    LldpV2DestAddressTableIndex
                      FROM LLDP-V2-TC-MIB
    systemGroup          FROM SNMPv2-MIB      -- [RFC3418]
;

ieee8021CnMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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: Norman Finn
    Postal: C/O IEEE 802.1 Working Group
             IEEE Standards Association
             445 Hoes Lane
             P.O. Box 1331
             Piscataway
             NJ 08855-1331
             USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG
"
DESCRIPTION
"Congestion notification module.
Unless otherwise indicated, the references in this MIB
module are to IEEE Std 802.1Q-2011."
```

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."

REVISION "201102270000Z" -- February 27, 2011

DESCRIPTION
"Minor edits to contact information etc. as part of
2011 revision of IEEE Std 802.1Q."

REVISION "200912180000Z" -- 12/18/2009 00:00GMT

DESCRIPTION
"Included in IEEE 802.1Qau-2010

Copyright (C) IEEE."
 ::= { iso(1) org(3) ieee(111)
 standards-association-numbers-series-standards (2)
 lan-man-stds (802) ieee802dot1 (1) ieee802dot1mibs (1) 18 }

ieee8021CnMIBObjects OBJECT IDENTIFIER ::= { ieee8021CnMib 1 }
ieee8021CnConformance OBJECT IDENTIFIER ::= { ieee8021CnMib 2 }

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

Ieee8021CnControlChoice ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"This controls what other object selects the CND defense mode and
the Congestion Notification Priority Value (CNPV) alternate
priority for a CNPV in an end station or bridge component, or
for a CNPV on a particular Port in an end station or bridge
component. It can take the following values:

cpcAdmin(1) The CND defense mode and alternate priority are
controlled by the administrative variables in the
same table entry as this object.
cpcAuto(2) This Port or all Ports' CND defense modes are
controlled automatically, as indicated by
ieee8021CnPortPriAutoDefenseMode, and the
alternate priority by ieee8021CnComPriAutoAltPri.
cpcComp(3) This CND defense mode and alternate priority are
both controlled by ieee8021CnPortPriTable.

"

REFERENCE
"IEEE 802.1Qau clauses 32.3.1, 32.4.1, Table 32-2"

SYNTAX INTEGER {
cpcAdmin (1),
cpcAuto (2),
cpcComp (3)
}

Ieee8021CnDefenseMode ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"For a given Congestion Notification Priority Value (CNPV), a
port can operate in one of four CND defense modes. The CND
defense mode determines whether congestion notification is
enabled or not, and if enabled, whether the port translates the
CNPV to a non-CNPV value on input, and whether the port removes

CN-TAGs on output.

cptDisabled(1)	The congestion notification capability is administratively disabled for this priority value and port. This priority is not a CNPV. The priority regeneration table controls the remapping of input frames on this port for this priority. CN-TAGs are neither added by an end station nor stripped by a bridge.
cptInterior(2)	On this port and for this CNPV, the priority parameters of input frames are not remapped, regardless of the priority regeneration table. CN-TAGs are not added by an end station, and are removed from frames before being output by a bridge.
cptInteriorReady(3)	On this port and for this CNPV, the priority parameters of input frames are not remapped, regardless of the priority regeneration table. CN-TAGs can be added by an end station, and are not removed from frames by a bridge.
cptEdge(4)	On this port and for this CNPV, the priority parameters of input frames are remapped to an alternate (non-CNPV) value, regardless of the priority regeneration table. CN-TAGs are not added by an end station, and are removed from frames before being output by a bridge. This mode is optional for an end station.
"	

REFERENCE

"IEEE 802.1Qau clause 32.1.1, 32.3.4, 32.4.2, 32.4.3, Table 32-2"

```
SYNTAX INTEGER {
    cptDisabled      (1),
    cptInterior     (2),
    cptInteriorReady (3),
    cptEdge          (4)
}
```

Ieee8021CnLldpChoice ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"Specifies how to determine what index value is to be used as the index to lldpDestAddressTable, the table of destination MAC addresses, for both the destination addresses on transmitted LLDPDUs and on received LLDPDUs, found in the LLDP-MIB, either:

cnlNone(1)	No LLDP Congestion Notification TLV is to carry Per-priority CNPV indicators or Per-priority Ready indicators on this Port for this priority (or all Ports and all priorities, as appropriate to the managed object).
cnlAdmin(2)	The administrative LLDP instance selector in the same table entry as this object governs which LLDP instance will carry the Per-priority CNPV indicators and Per-priority Ready indicators for this priority in its Congestion Notification TLV on this Port (or all Ports and all

```

priorities, as appropriate to the managed
object).
cnlComponent(3) ieee8021CnComPriLldpInstanceSelector governs
LLDP instance selection for this Port and
priority.

"
REFERENCE
"IEEE 802.1Qau clause 32.3.6, 32.4.4, Table 32-1"
SYNTAX INTEGER {
    cnlNone      (1),
    cnlAdmin     (2),
    cnlComponent (3)
}

-- *****
-- The Global Table. This group will contain the MIB objects that
-- apply to the whole bridge component or end station.
-- *****

ieee8021CnGlobalTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021CnGlobalEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table containing the global variables for each component of
a bridge or for an end station. A value of 1 is used in a
bridge or end station that does not have multiple components.

The contents of this table SHALL be maintained across a restart
of the system.
"
REFERENCE
"802.1Qau clause 12.21.1"
 ::= { ieee8021CnMIBObjects 1 }

ieee8021CnGlobalEntry OBJECT-TYPE
SYNTAX      Ieee8021CnGlobalEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A list of objects pertaining to a whole bridge component or
end station.
"
INDEX { ieee8021CnGlobalComponentId }
 ::= { ieee8021CnGlobalTable 1 }

Ieee8021CnGlobalEntry ::= SEQUENCE {
    ieee8021CnGlobalComponentId IEEE8021PbbComponentIdentifier,
    ieee8021CnGlobalMasterEnable          TruthValue,
    ieee8021CnGlobalCnmTransmitPriority IEEE8021PriorityValue,
    ieee8021CnGlobalDiscardedFrames       Counter64
}

ieee8021CnGlobalComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current

```

DESCRIPTION
"The bridge component within the system to which the information in this ieee8021CnGlobalEntry 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.
"
 ::= { ieee8021CnGlobalEntry 1 }

ieee8021CnGlobalMasterEnable OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The state of the congestion notification feature on this bridge component or system. If true, Congestion notification is enabled, and if false, congestion notification is disabled.
"
REFERENCE
"802.1Qau clause 32.2.1"
 ::= { ieee8021CnGlobalEntry 2 }

ieee8021CnGlobalCnmTransmitPriority OBJECT-TYPE
SYNTAX IEEE8021PriorityValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The priority to use for all Congestion Notification Messages transmitted by this bridge component or end station.
"
REFERENCE
"802.1Qau clause 32.2.2"
 ::= { ieee8021CnGlobalEntry 3 }

ieee8021CnGlobalDiscardedFrames OBJECT-TYPE
SYNTAX Counter64
UNITS "frames"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of frames discarded from full CP queues, in spite of the efforts of congestion notification to avoid discards.

This object is incremented whenever any of the ieee8021CnCpDiscardedFrames objects on any Port or priority in this same component are incremented.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any).
"
REFERENCE
"802.1Qau clause 32.2.3"
 ::= { ieee8021CnGlobalEntry 4 }

-- *****

```
-- The CCF Errored Port Table. One per bridge component or end station.
-- Scanning this table reveals which Interfaces at which priorities
-- have an Alternate Priority value that is a
-- Congestion Notification Priority Value.
-- ****
```

```
ieee8021CnErroredPortTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021CnErroredPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "There is one Errored Port Table per bridge component or end
         station. It permits the retrieval of information about which
         interfaces have congestion notification configuration errors,
         namely, those specifying an alternate priority that is a CNPV.
        "
    REFERENCE
        "802.1Qau clause 32.2.4"
    ::= { ieee8021CnMIBObjects 2 }
```

```
ieee8021CnErroredPortEntry OBJECT-TYPE
    SYNTAX      Ieee8021CnErroredPortEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of interfaces whose ieee8021CnComPriAlternatePriority
         and/or ieee8021CnPortPriAlternatePriority specify a priority
         value that is a Congestion Notification Priority Value.
        "
    REFERENCE
        "802.1Qau clause 32.2.4"
    INDEX { ieee8021CnEpComponentId,
            ieee8021CnEpPriority,
            ieee8021CnEpIfIndex}
    ::= { ieee8021CnErroredPortTable 1 }
```

```
Ieee8021CnErroredPortEntry ::= SEQUENCE {
    ieee8021CnEpComponentId      IEEE8021PbbComponentIdentifier,
    ieee8021CnEpPriority         IEEE8021PriorityValue,
    ieee8021CnEpIfIndex          InterfaceIndex
}
```

```
ieee8021CnEpComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The bridge component within the system to which the information
         in this ieee8021CnErroredPortEntry 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
        "802.1Qau clause 32.2.4"
    ::= { ieee8021CnErroredPortEntry 1 }
```

```
ieee8021CnEpPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
```

```
STATUS      current
DESCRIPTION
    "The priority value whose alternate priority is misconfigured.
"
REFERENCE
    "802.1Qau clause 32.2.4"
::= { ieee8021CnErroredPortEntry 2 }

ieee8021CnEpIfIndex OBJECT-TYPE
SYNTAX      InterfaceIndex
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "This object represents the Bridge Port or aggregated port
     on which the congestion notification alternate priority is
     misconfigured.
Upon a restart of the system, the system SHALL, if necessary,
change the value of this variable so that it references the row
in the ifXTable with the same value of ifAlias that it
referenced before the system restart. If no such row exists,
then the system SHALL delete this row in the
ieee8021CnErroredPortTable.
"
REFERENCE
    "802.1Qau clause 32.2.4"
::= { ieee8021CnErroredPortEntry 3 }

-- *****
-- The CCF Per-component per-priority table. One table per bridge
-- component or end station, one entry per
-- Congestion Notification Priority Value
-- *****

ieee8021CnComptPriTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021CnComptPriEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "Each row in this table supplies default values for one
     Congestion Notification Priority Value for a whole bridge
     component or end station.

Creating a row in this table makes the priority value of
ieee8021CnComPriPriority a
Congestion Notification Priority Value.
Deleting a row in this table makes the value in the deleted
ieee8021CnComPriPriority no longer a
Congestion Notification Priority Value.

A system SHALL NOT allow eight rows in this table
to be created with the same value of
ieee8021CnComPriComponentId; see the description of
ieee8021CnComPriRowStatus.

The contents of this table SHALL be maintained across a restart
of the system.
"
REFERENCE
```

```

ieee8021CnCompntPriEntry OBJECT-TYPE
    SYNTAX      Ieee8021CnCompntPriEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "One entry per Congestion Notification Priority Value per
         bridge component or end station.
        "
    REFERENCE
        "802.1Qau clause 12.21.2, 12.21.2.1, 12.21.2.2"
    INDEX { ieee8021CnComPriComponentId,
            ieee8021CnComPriPriority }
    ::= { ieee8021CnCompntPriTable 1 }

Ieee8021CnCompntPriEntry ::= SEQUENCE {
    ieee8021CnComPriComponentId      IEEE8021PbbComponentIdentifier,
    ieee8021CnComPriPriority        IEEE8021PriorityValue,
    ieee8021CnComPriDefModeChoice   Ieee8021CnControlChoice,
    ieee8021CnComPriAlternatePriority IEEE8021PriorityValue,
    ieee8021CnComPriAutoAltPri     IEEE8021PriorityValue,
    ieee8021CnComPriAdminDefenseMode Ieee8021CnDefenseMode,
    ieee8021CnComPriCreation       INTEGER,
    ieee8021CnComPriLldpInstanceChoice Ieee8021CnLldpChoice,
    ieee8021CnComPriLldpInstanceSelector LldpV2DestAddressTableIndex,
    ieee8021CnComPriRowStatus       RowStatus
}

ieee8021CnComPriComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The bridge component within the system to which the information
         in this ieee8021CnCompntPriEntry 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
        "802.1Qau clause 12.21.2, 12.21.2.1, 12.21.2.2"
    ::= { ieee8021CnCompntPriEntry 1 }

ieee8021CnComPriPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Congestion Notification Priority Value for which this
         row supplies default values.
        "
    REFERENCE
        "802.1Qau clauses 12.21.2"
    ::= { ieee8021CnCompntPriEntry 2 }

ieee8021CnComPriDefModeChoice OBJECT-TYPE
    SYNTAX      Ieee8021CnControlChoice {
        cpcAdmin      (1),

```

```
        cpcAuto      (2)
    }
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "Specifies how the default CND defense mode and alternate
     priority for this Congestion Notification Priority Value on all
     ports on this bridge component or end station are to be chosen,
     either:
        cpcAdmin(1) Default CND defense mode is chosen by
                      ieee8021CnComPriAdminDefenseMode, and alternate
                      priority by ieee8021CnComPriAlternatePriority.
        cpcAuto(2)   Default CND defense mode is chosen by
                      ieee8021CnPortPriAutoDefenseMode, and alternate
                      priority by ieee8021CnComPriAutoAltPri.

    This variable can be overridden by
    ieee8021CnPortPriDefModeChoice.
"
REFERENCE
    "802.1Qau clause 32.1.3, 32.3.1"
DEFVAL { cpcAuto }
::= { ieee8021CnCompntPriEntry 3 }

ieee8021CnComPriAlternatePriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The Congestion Notification Priority Value to which an
     incoming frame is to be mapped, in spite of what the
     Priority Regereration Table says, if 1) Congestion
     Notification is enabled and 2) the CND defense mode of the
     port is cptEdge.

    Deleting a row in this table does not alter the value of any
    other row's ieee8021CnComPriAlternatePriority.
"
REFERENCE
    "802.1Qau clauses 32.3.2"
DEFVAL { 0 }
::= { ieee8021CnCompntPriEntry 4 }

ieee8021CnComPriAutoAltPri OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The Congestion Notification Priority Value to which an
     incoming frame can be mapped, in spite of what the
     Priority Regereration Table says, if 1) Congestion
     Notification is enabled, 2) the CND defense mode of the
     port is cptEdge, and 3) ieee8021CnComPriDefModeChoice
     contains the value cpcAuto (2).

    The value of this object is the next lower priority value
    than this row's ieee8021CnComPriPriority that is not a CNPV,
    or the next higher non-CNPV, if all lower values are CNPVs.
```

The value of this object, and any consequent priority regeneration, is automatically updated by the managed system whenever a row in the ieee8021CnComptnPriTable is created or deleted. The value of this object is not dependent upon whether congestion notification is enabled or disabled for any priority or for the whole bridge component or end station; it depends only upon whether the ieee8021CnComptnPriTable row exists.

"

REFERENCE

"802.1Qau clauses 32.3.3"

::= { ieee8021CnComptnPriEntry 5 }

ieee8021CnComPriAdminDefenseMode OBJECT-TYPE

SYNTAX Ieee8021CnDefenseMode

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The default CND defense mode for this Congestion Notification Priority Value on all ports on this bridge component or end station.

This variable can be overridden by ieee8021CnPortPriAdminDefenseMode.

"

REFERENCE

"802.1Qau clause 32.1.3, 32.3.4"

DEFVAL { cptInterior }

::= { ieee8021CnComptnPriEntry 6 }

ieee8021CnComPriCreation OBJECT-TYPE

SYNTAX INTEGER {
 cncpAutoEnable (1),
 cncpAutoDisable (2)
 }

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The default value for ieee8021CnComPriDefModeChoice for newly-created entries in the ieee8021CnPortPriTable:

cncpAutoEnable (1) Newly-created
 ieee8021CnPortPriDefModeChoice
 objects take the value cpcComp (3).
 cncpAutoDisable (2) Newly-created
 ieee8021CnPortPriDefModeChoice
 objects take the value cpcAdmin (1).

"

REFERENCE

"802.1Qau clause 32.3.5"

DEFVAL { cncpAutoEnable }

::= { ieee8021CnComptnPriEntry 7 }

ieee8021CnComPriLldpInstanceChoice OBJECT-TYPE

SYNTAX Ieee8021CnLldpChoice {
 cnlNone (1),
 cnlAdmin (2)
 }

```
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
  "Specifies whether or not the default value for all Ports is to
   send and receive the Congestion Notification TLV in LLDPDUs,
   either:
    cnlNone(1)  Do not send Congestion Notification TLVs, and
                 ignore them on receipt.
    cnlAdmin(2)  Use the LLDP instance selected by
                 ieee8021CnComPriLldpInstanceSelector to send and
                 receive the Congestion Notification TLV.

  This object can be overridden by
  ieee8021CnPortPriLldpInstanceChoice.
  "
REFERENCE
  "802.1Qau clause 32.1.3, 32.3.6"
DEFVAL { cnlAdmin }
::= { ieee8021CnCompntPriEntry 8 }

ieee8021CnComPriLldpInstanceSelector OBJECT-TYPE
  SYNTAX      LldpV2DestAddressTableIndex
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
  "Specifies a default value for which LLDP instance is to be
   used to provide the information for automatic configuration
   of ports' CND defense modes (ieee8021CnPortPriAutoDefenseMode).

  This object is ignored by the managed system if
  ieee8021CnComPriLldpInstanceChoice contains the value cnlNone
  (1).

  This object can be overridden by
  ieee8021CnPortPriLldpInstanceChoice and
  ieee8021CnPortPriLldpInstanceChoice.
  "
REFERENCE
  "802.1Qau clause 32.1.3, 32.3.7"
DEFVAL { 1 }
::= { ieee8021CnCompntPriEntry 9 }

ieee8021CnComPriRowStatus OBJECT-TYPE
  SYNTAX      RowStatus
  MAX-ACCESS  read-create
  STATUS      current
  DESCRIPTION
  "This object indicates the status of an entry, and is used
   to create/delete entries.

  A system SHALL NOT permit eight ieee8021CnComPriRowStatus
  objects, all with the same value of ieee8021CnComPriComponentId,
  to have the value active(1). An attempt to create or activate
  a row when there are already seven active rows SHALL result in
  that eighth row's ieee8021CnComPriRowStatus having the value
  notReady(3), and the return of an inconsistentValue error.
  "
REFERENCE
  "802.1Qau clause 30.4"
```

```

 ::= { ieee8021CnCompntPriEntry 10 }

-- ****
-- The CCF Per-component per-interface per-priority table. One table
-- per end station or bridge component. One entry per interface per
-- priority value, but only for priority values that are
-- Congestion Notification Priority Values (CNPVs). Controls
-- a CNPV on a specific port when the default values in
-- ieee8021CnCompntPriTable need to be overridden.
-- ****

ieee8021CnPortPriTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF Ieee8021CnPortPriEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "Each row in this table supplies values for one port's
     Congestion Notification Priority Value (CNPV).

     Creating an entry in ieee8021CnCompntPriTable creates this
     entry, with the default values, on all ports in the bridge
     component or end station. Deleting an entry in
     ieee8021CnCompntPriTable deletes this ieee8021CnCompntPriEntry
     on all ports in the bridge component or end station.

     The contents of this table SHALL be maintained across a restart
     of the system, except as noted in the description of
     ieee8021CnPortPriIfIndex.

    "
  REFERENCE
    "802.1Qau clause 12.21.3"
 ::= { ieee8021CnMIBObjects 4 }

ieee8021CnPortPriEntry OBJECT-TYPE
  SYNTAX      Ieee8021CnPortPriEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "One entry per port per Congestion Notification Priority Value
     per bridge component or end station.

    "
  REFERENCE
    "802.1Qau clause 12.21.3"
  INDEX { ieee8021CnPortPriComponentId,
           ieee8021CnPortPriority,
           ieee8021CnPortPriIfIndex }
 ::= { ieee8021CnPortPriTable 1 }

Ieee8021CnPortPriEntry ::= SEQUENCE {
    ieee8021CnPortPriComponentId    IEEE8021PbbComponentIdentifier,
    ieee8021CnPortPriority         IEEE8021PriorityValue,
    ieee8021CnPortPriIfIndex       InterfaceIndex,
    ieee8021CnPortPriDefModeChoice IEEE8021CnControlChoice,
    ieee8021CnPortPriAdminDefenseMode IEEE8021CnDefenseMode,
    ieee8021CnPortPriAutoDefenseMode IEEE8021CnDefenseMode,
    ieee8021CnPortPriLldpInstanceChoice IEEE8021CnLldpChoice,
    ieee8021CnPortPriLldpInstanceSelector
                                LldpV2DestAddressTableIndex,
}

```

```
        ieee8021CnPortPriAlternatePriority    IEEE8021PriorityValue
    }

ieee8021CnPortPriComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The bridge component within the system to which the information
         in this ieee8021CnPortPriEntry 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
        "802.1Qau clause 12.21.3"
    ::= { ieee8021CnPortPriEntry 1 }

ieee8021CnPortPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The Congestion Notification Priority Value for which
         this row supplies default values.
        "
    REFERENCE
        "802.1Qau clauses 12.21.3"
    ::= { ieee8021CnPortPriEntry 2 }

ieee8021CnPortPriIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object represents the port or aggregated port
         to which the entry applies.

Upon a restart of the system, the system SHALL, if necessary,
change the value of this object, and rearrange the order of the
ieee8021CnPortPriTable, so that the value in
ieee8021CnPortPriIfIndex references the row in the ifXTable
with the same value for ifAlias that it referenced before the
system restart. If no such entry exists in the ifXTable, then
the system SHALL delete the row in the ieee8021CnPortPriTable.
        "
    REFERENCE
        "802.1Qau clause 12.21.3"
    ::= { ieee8021CnPortPriEntry 3 }

ieee8021CnPortPriDefModeChoice OBJECT-TYPE
    SYNTAX  Ieee8021CnControlChoice
    MAX-ACCESS  read-write
    STATUS  current
    DESCRIPTION
        "This object determines how the CND defense mode and alternate
         priority value of this port for this CNPV is to be selected,
         either:
            cpcAdmin(1) CND defense mode is controlled by

```

ieee8021CnPortPriAdminDefenseMode, and alternate priority by ieee8021CnPortPriAlternatePriority.
 cpcAuto(2) CND defense mode is controlled by ieee8021CnPortPriAutoDefenseMode and alternate priority by ieee8021CnComPriAlternatePriority.
 cpcComp(3) CND defense mode and alternate priority are controlled by ieee8021CnComPriDefModeChoice.

This variable can override ieee8021CnComPriDefModeChoice.

"

REFERENCE

"IEEE 802.1Qau clause 32.1.3, 32.4.1"
 DEFVAL { cpcComp }
 ::= { ieee8021CnPortPriEntry 4 }

ieee8021CnPortPriAdminDefenseMode OBJECT-TYPE
SYNTAX Ieee8021CnDefenseMode
MAX-ACCESS read-write
STATUS current
DESCRIPTION
 "This object indicates the operator's choice for the CND defense mode in which this port is to operate for this CNPV whenever ieee8021CnPortPriDefModeChoice has the value cpcAdmin(1)."

This variable can override ieee8021CnComPriDefModeChoice.

"

REFERENCE

"IEEE 802.1Qau clause 32.1.3, 32.4.1"
 DEFVAL { cptDisabled }
 ::= { ieee8021CnPortPriEntry 5 }

ieee8021CnPortPriAutoDefenseMode OBJECT-TYPE
SYNTAX Ieee8021CnDefenseMode {
 cptInterior (2),
 cptInteriorReady (3),
 cptEdge (4)
}
MAX-ACCESS read-only
STATUS current
DESCRIPTION
 "This object indicates in which the CND defense mode this port would operate for this CNPV as determined by the LLDP Congestion Notification TLV."
 "
REFERENCE
 "IEEE 802.1Qau clause 32.4.3"
 ::= { ieee8021CnPortPriEntry 6 }

ieee8021CnPortPriLldpInstanceChoice OBJECT-TYPE
SYNTAX Ieee8021CnLldpChoice
MAX-ACCESS read-write
STATUS current
DESCRIPTION
 "Specifies how to determine the LLDP instance to be used for the Congestion Notification TLV, either:
 cnlNone(1) No LLDP Congestion Notification TLV is to carry Per-priority CNPV indicators or

```
Per-priority Ready indicators on this Port
for this priority.
cnlAdmin(2) ieee8021CnPortPriLldpInstanceSelector
governs which LLDP instance is to carry
Per-priority CNPV indicators and
Per-priority Ready indicators for this
priority in its Congestion Notification TLV
on this Port
cnlComponent(3) ieee8021CnComPriLldpInstanceChoice
governs LLDP instance selection for this
Port and priority.

This object can override ieee8021CnComPriLldpInstanceChoice.
"
REFERENCE
"802.1Qau clause 32.1.3, 32.4.4"
DEFVAL { cnlComponent }
 ::= { ieee8021CnPortPriEntry 7 }

ieee8021CnPortPriLldpInstanceSelector OBJECT-TYPE
SYNTAX      LldpV2DestAddressTableIndex
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"This object determines which LLDP instance selector, if any,
is used for automatic determination of the CND defense mode for
this port and CNPV.

This object can override ieee8021CnComPriLldpInstanceSelector.
"
REFERENCE
"802.1Qau clause 32.1.3, 32.4.5"
DEFVAL { 3 }
 ::= { ieee8021CnPortPriEntry 8 }

ieee8021CnPortPriAlternatePriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"The Congestion Notification Priority Value to which an
incoming frame is to be mapped, in spite of what the
Priority Regereration Table says, if 1) Congestion
Notification is enabled and 2) the port is acting in the
cptEdge (4) CND defense mode.

This object is ignored unless ieee8021CnPortPriDefModeChoice
contains the value cpcAdmin (1).
"
REFERENCE
"802.1Qau clause 32.4.6"
DEFVAL { 0 }
 ::= { ieee8021CnPortPriEntry 9 }

-- ****
-- The CCF per-CP table. One table per end station or bridge component.
-- One entry per Congestion Point. An entry in this table controls one
```

```
-- Congestion Point (CP).
-- ****

ieee8021CnCpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021CnCpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Each row in this table supplies values for one
         Congestion Point (CP)."

    This table is indexed by component, port (interface), and
    an arbitrary CP index. This arbitrary CP index is not
    necessarily the Congestion Point Identifier (CPID) carried in
    Congestion Notification Messages (CNMs).

    Creating an entry in ieee8021CnCompntPriTable can create an
    entry in this table, with the default values, on all ports in
    the bridge component or end station. Because more than one
    Congestion Notification Priority Value (CNPV) can flow
    through a single CP, the creation of an entry in
    ieee8021CnCompntPriTable does not necessarily create a new
    entry in this table. An end station can have more than one
    CP for the same CNPV, so creating an entry in
    ieee8021CnCompntPriTable can create multiple entries in this
    table.

    Because each port in a bridge component or end station can have
    a different relationship between CNPVs and CPs, the entries
    created or deleted on each port can be different.

    Deleting the last entry in ieee8021CnCompntPriTable for a
    CNPV passing through the CP controlled by this entry deletes
    the entry on some or all of the ports in the bridge component
    or end station.

    Because each port in a bridge component or end station can have
    a different relationship between CNPVs and CPs, the entries
    created or deleted on each port can be different.

    The relationship between ieee8021CnCpIndex
    values and CPs is an implementation dependent matter.

    The contents of this table SHALL be maintained across a restart
    of the system, except as noted in the description of
    ieee8021CnCpIfIndex.

    "
    REFERENCE
        "802.1Qau clause 12.21.4"
    ::= { ieee8021CnMIBObjects 5 }

ieee8021CnCpEntry OBJECT-TYPE
    SYNTAX      Ieee8021CnCpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry in the Congestion Point table controls a single
         Congestion Point on a port in a bridge component or end station.
        "

```

```
REFERENCE
    "802.1Qau clause 12.21.4"
INDEX { ieee8021CnCpComponentId,
         ieee8021CnCpIfIndex,
         ieee8021CnCpIndex }
 ::= { ieee8021CnCpTable 1 }

Ieee8021CnCpEntry ::= SEQUENCE {
    ieee8021CnCpComponentId      IEEE8021PbbComponentIdentifier,
    ieee8021CnCpIfIndex          InterfaceIndex,
    ieee8021CnCpIndex            Unsigned32,
    ieee8021CnCpPriority         IEEE8021PriorityValue,
    ieee8021CnCpMacAddress       MacAddress,
    ieee8021CnCpIdentifier       OCTET STRING,
    ieee8021CnCpQueueSizeSetPoint Unsigned32,
    ieee8021CnCpFeedbackWeight   Integer32,
    ieee8021CnCpMinSampleBase    Unsigned32,
    ieee8021CnCpDiscardedFrames Counter64,
    ieee8021CnCpTransmittedFrames Counter64,
    ieee8021CnCpTransmittedCnms Counter64,
    ieee8021CnCpMinHeaderOctets Unsigned32
}

ieee8021CnCpComponentId OBJECT-TYPE
SYNTAX      IEEE8021PbbComponentIdentifier
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The bridge component within the system to which the information
     in this ieee8021CnCpEntry 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
    "802.1Qau clause 12.21.4"
 ::= { ieee8021CnCpEntry 1 }

ieee8021CnCpIfIndex OBJECT-TYPE
SYNTAX      InterfaceIndex
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "This object represents the port or aggregated port
     to which the entry applies.

Upon a restart of the system, the system SHALL, if necessary,
change the value of this object, and rearrange the order of the
ieee8021CnCpTable, so that the value in ieee8021CnCpIfIndex
references the row in the ifXTable with the same value for
ifAlias that it referenced before the system restart. If no
such entry exists in the ifXTable, then the system SHALL delete
the row in the ieee8021CnCpTable.
"
REFERENCE
    "802.1Qau clause 12.21.4"
 ::= { ieee8021CnCpEntry 2 }

ieee8021CnCpIndex OBJECT-TYPE
SYNTAX      Unsigned32 (1..4096)
```

```

MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
  "This object is an arbitrary integer indexing the entries in
  this table among the entries for the same component and
  interface. In a system that supports no more than one
  Congestion Point per priority per interface, ieee8021CnCpIndex
  SHALL be equal to the lowest numerical
  Congestion Notification Priority Value served by this
  Congestion Point. Otherwise, it SHOULD be a small integer
  value.
"
REFERENCE
  "802.1Qau clause 12.21.4"
 ::= { ieee8021CnCpEntry 3 }

ieee8021CnCpPriority OBJECT-TYPE
SYNTAX      IEEE8021PriorityValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "This object indicates the lowest numerical
  Congestion Notification Priority Value that this entry's
  Congestion Point serves.
"
REFERENCE
  "802.1Qau clause 12.21.4"
 ::= { ieee8021CnCpEntry 4 }

ieee8021CnCpMacAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "This object indicates the MAC address used as the source
  address in Congestion Notification Message transmitted
  by this Congestion Point.
"
REFERENCE
  "802.1Qau clause 32.8.1"
 ::= { ieee8021CnCpEntry 5 }

ieee8021CnCpIdentifier OBJECT-TYPE
SYNTAX      OCTET STRING (SIZE(8))
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
  "This object indicates the Congestion Point Identifier (CPIID)
  transmitted in Congestion Notification Message by this
  Congestion Point.

  It is not specified whether the CPIID reported in a CNM by a CP
  that serves multiple CNPVs does or does not have the same value
  for its different CNPVs.
"
REFERENCE
  "802.1Qau clause 32.8.2"
 ::= { ieee8021CnCpEntry 6 }

```

```
ieee8021CnCpQueueSizeSetPoint OBJECT-TYPE
    SYNTAX      Unsigned32 (100..4294967295)
    UNITS      "octets"
    MAX-ACCESS  read-write
    STATUS     current
    DESCRIPTION
        "This object is the set point for the queue managed by this
         Congestion Point (CP). Congestion Notification Messages are
         transmitted to the sources of frames queued in this CP's
         queue in order to keep the total number of octets stored in
         the queue at this set point.
        "
    REFERENCE
        "802.1Qau clause 30.2, 32.8.3"
    DEFVAL { 26000 }
    ::= { ieee8021CnCpEntry 7 }

ieee8021CnCpFeedbackWeight OBJECT-TYPE
    SYNTAX      Integer32 (-10..10)
    MAX-ACCESS  read-write
    STATUS     current
    DESCRIPTION
        "This object controls the weight (cpW) change in queue length
         in the calculation of cpFb when the Congestion Point is
         generating a Congestion Notification Message.

        The weight cpW is equal to two to the power of this object.
        Thus, if this object contains a -1, cpW = 1/2.
        "
    REFERENCE
        "802.1Qau clause 32.8.6"
    DEFVAL { 1 }
    ::= { ieee8021CnCpEntry 8 }

ieee8021CnCpMinSampleBase OBJECT-TYPE
    SYNTAX      Unsigned32 (10000..4294967295)
    UNITS      "octets"
    MAX-ACCESS  read-write
    STATUS     current
    DESCRIPTION
        "This object determines the minimum number of octets to
         enqueue in the Congestion Point's queue between transmissions
         of Congestion Notification Messages.
        "
    REFERENCE
        "802.1Qau clause 32.8.11"
    DEFVAL { 150000 }
    ::= { ieee8021CnCpEntry 9 }

ieee8021CnCpDiscardedFrames OBJECT-TYPE
    SYNTAX      Counter64
    UNITS      "frames"
    MAX-ACCESS  read-only
    STATUS     current
    DESCRIPTION
        "The number of data frames discarded by the queue controlled
         by this Congestion Point due to queue congestion.

        Discontinuities in the value of this counter can occur
```

at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any).

"

REFERENCE

"802.1Qau clause 32.8.12"
 ::= { ieee8021CnCpEntry 10 }

ieee8021CnCpTransmittedFrames OBJECT-TYPE

SYNTAX Counter64

UNITS "frames"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of data frames passed on to the queue controlled by this Congestion Point that were not discarded due to queue congestion.

Discontinuities in the value of this counter can occur at re-initialization of the management system, and at other times as indicated by the value of ifCounterDiscontinuityTime object of the associated interface (if any).

"

REFERENCE

"802.1Qau clause 32.8.13"
 ::= { ieee8021CnCpEntry 11 }

ieee8021CnCpTransmittedCnms OBJECT-TYPE

SYNTAX Counter64

UNITS "frames"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

The number of Congestion Notification Message transmitted by this Congestion Point.

"

REFERENCE

"802.1Qau clause 32.8.14"
 ::= { ieee8021CnCpEntry 12 }

ieee8021CnCpMinHeaderOctets OBJECT-TYPE

SYNTAX Unsigned32 (0..64)

UNITS "octets"

MAX-ACCESS read-write

STATUS current

DESCRIPTION

Specifies the minimum number of octets to be returned in a Congestion Notification Message from the mac_service_data_unit of the data frame that triggered transmission of the CNM. If the mac_service_data_unit has fewer octets than the value of this object, then all of the mac_service_data_unit is returned

```
    in the CNM.  
"  
REFERENCE  
    "802.1Qau clause 32.8.15, 32.9.4 k)"  
DEFVAL { 0 }  
 ::= { ieee8021CnCpEntry 13 }  
  
-- ****  
-- The CPID to {Interface, CP index} table. One table per system.  
-- One entry per CPID.  
-- ****  
  
ieee8021CnCpidToInterfaceTable OBJECT-TYPE  
SYNTAX      SEQUENCE OF Ieee8021CnCpidToInterfaceEntry  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "This table allows the network manager to obtain the  
    interface index and CP index needed to access an entry in  
    the ieee8021CnCpTable, given a Congestion Point Identifier  
    (CPID) received a Congestion Notification Messages (CNMs).  
  
Upon a restart of the system, the system SHALL, if necessary,  
update this table to be consistent with the ieee8021CnCpTable.  
"  
REFERENCE  
    "802.1Qau clause 17.2.13, 12.21.4, 32.8.2"  
 ::= { ieee8021CnMIBObjects 6 }  
  
ieee8021CnCpidToInterfaceEntry OBJECT-TYPE  
SYNTAX      Ieee8021CnCpidToInterfaceEntry  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "An entry in the ieee8021CnCpidToInterfaceTable. Translates  
    a Congestion Point Identifier to a component identifier,  
    interface index, and CP index  
"  
REFERENCE  
    "802.1Qau clause 17.2.13"  
INDEX { ieee8021CnCpidToIfCpid }  
 ::= { ieee8021CnCpidToInterfaceTable 1 }  
  
Ieee8021CnCpidToInterfaceEntry ::= SEQUENCE {  
    ieee8021CnCpidToIfCpid          OCTET STRING,  
    ieee8021CnCpidToIfComponentId   IEEE8021PbbComponentIdentifier,  
    ieee8021CnCpidToIfIfIndex       InterfaceIndex,  
    ieee8021CnCpidToIfCpIndex       Unsigned32  
}  
  
ieee8021CnCpidToIfCpid OBJECT-TYPE  
SYNTAX      OCTET STRING (SIZE(8))  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "This object is the Congestion Point Identifier (CPID)  
    transmitted in Congestion Notification Message by a
```

```

        Congestion Point residing in this bridge component or
        end station.
    "
REFERENCE
    "802.1Qau clause 17.2.13, 32.8.2"
    ::= { ieee8021CnCpidToInterfaceEntry 1 }

ieee8021CnCpidToIfComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The bridge component within the system to which the information
        in this ieee8021CnCpidToInterfaceEntry 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
    "802.1Qau clause 17.2.13"
    ::= { ieee8021CnCpidToInterfaceEntry 2 }

ieee8021CnCpidToIfIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the interface on which the selected
        Congestion Point resides. This value can be used, along
        with ieee8021CnCpidToIfCpIndex, to find the Congestion Point
        in the ieee8021CnCpTable.
    "
REFERENCE
    "802.1Qau clause 17.2.13"
    ::= { ieee8021CnCpidToInterfaceEntry 3 }

ieee8021CnCpidToIfCpIndex OBJECT-TYPE
    SYNTAX      Unsigned32 (1..4096)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object indicates the Congestion Point's index on the
        interface on which the selected Congestion Point resides.
        This value can be used, along with ieee8021CnCpidToIfIfIndex,
        to find the Congestion Point in the ieee8021CnCpTable.
    "
REFERENCE
    "802.1Qau clause 17.2.13"
    ::= { ieee8021CnCpidToInterfaceEntry 4 }

```

```
-- ****
-- The Reaction Point Port table. One table per end station. One
-- entry per Port per CNPV.
-- ****
```

```
ieee8021CnRpPortPriTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021CnRpPortPriEntry
    MAX-ACCESS  not-accessible
```

```
STATUS      current
DESCRIPTION
  "Each row in this table supplies values for all of the
  Reaction Points (RPs) on one Port and one priority of an end
  station or bridge component. This table is indexed by
  component, port (interface), and priority.

  Creating an entry in ieee8021CnCompntPriTable can create an
  entry in this table, with the default values, on all ports
  in the end station.

  Deleting the an entry in ieee8021CnCompntPriTable for a
  CNPV passing through the RP controlled by this entry deletes
  entries on some or all of the ports in the end station.

  The contents of this table SHALL be maintained across a restart
  of the system, except as noted in the description of
  ieee8021CnRpPortPriIfIndex.

  "
REFERENCE
  "802.1Qau clause 12.21.5"
::= { ieee8021CnMIBObjects 7 }

ieee8021CnRpPortPriEntry OBJECT-TYPE
  SYNTAX      Ieee8021CnRpPortPriEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "An entry in the Reaction Point table controls all of the
    Reaction Points on a port in an end station that share the same
    priority value.
    "
  REFERENCE
    "802.1Qau clause 12.21.5"
  INDEX { ieee8021CnRpPortPriComponentId, ieee8021CnRpPortPriPriority,
          ieee8021CnRpPortPriIfIndex
        }
  ::= { ieee8021CnRpPortPriTable 1 }

Ieee8021CnRpPortPriEntry ::= SEQUENCE {
  ieee8021CnRpPortPriComponentId    IEEE8021PbbComponentIdentifier,
  ieee8021CnRpPortPriPriority      IEEE8021PriorityValue,
  ieee8021CnRpPortPriIfIndex       InterfaceIndex,
  ieee8021CnRpPortPriMaxRps      Unsigned32,
  ieee8021CnRpPortPriCreatedRps   Counter32,
  ieee8021CnRpPortPriCentiseconds Counter64
}

ieee8021CnRpPortPriComponentId OBJECT-TYPE
  SYNTAX      IEEE8021PbbComponentIdentifier
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The bridge component within the system to which the information
    in this ieee8021CnRpGroupEntry 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.
    "
  ::= { ieee8021CnRpPortPriEntry 1 }
```

```

ieee8021CnRpPortPriPriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object indicates the lowest numerical
         Congestion Notification Priority Value that this entry's
         Reaction Point serves.
        "
    REFERENCE
        "802.1Qau clause 12.21.5"
    ::= { ieee8021CnRpPortPriEntry 2 }

```

```

ieee8021CnRpPortPriIfIndex OBJECT-TYPE
    SYNTAX      InterfaceIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object indicates the interface on which the selected
         Reaction Points reside.

```

Upon a restart of the system, the system SHALL, if necessary, change the value of this object, and rearrange the order of the ieee8021CnRpPortPriTable, so that the value in ieee8021CnRpPortPriIfIndex references the row in the ifXTable with the same value for ifAlias that it referenced before the system restart. If no such entry exists in the ifXTable, then the system SHALL delete the row in the ieee8021CnRpPortPriTable.

```

    "
    REFERENCE
        "802.1Qau clause 12.21.5"
    ::= { ieee8021CnRpPortPriEntry 3 }

```

```

ieee8021CnRpPortPriMaxRps OBJECT-TYPE
    SYNTAX      Unsigned32 (1..100)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "An integer controlling the maximum number of Reaction Points
         allowed for this CNPV on this Port. An end station SHALL
         not create more than this many Reaction Point on this Port,
         but it MAY create fewer.
        "
    REFERENCE
        "802.1Qau clause 32.10.1"
    DEFVAL { 1 }
    ::= { ieee8021CnRpPortPriEntry 4 }

```

```

ieee8021CnRpPortPriCreatedRps OBJECT-TYPE
    SYNTAX      Counter32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object returns the number of times any of the
         Reaction Points (RPs) controlled by this entry has had
         its rpEnabled variable set TRUE by the reception of a
         Congestion Notification Message.

```

Dividing the change in ieee8021CnRpPortPriCentiseconds by the change in this object over a time interval yields the average lifetime of an active RP during that interval.

"

REFERENCE

"802.1Qau clause 32.10.2, 32.10.3, 32.13.1"

::= { ieee8021CnRpPortPriEntry 5 }

ieee8021CnRpPortPriCentiseconds OBJECT-TYPE

SYNTAX Counter64

UNITS "centiseconds"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"This object returns the total number of centi-seconds that any of the Reaction Points (RPs) controlled by this entry has had its rpEnabled variable in the TRUE state. That is, once each centi-second, this counter is incremented by the number of RPs this entry controls that are actively rate limiting output frames.

Dividing the change in this object over a time interval by the length of the interval yields the average number of RPs active over that interval. Dividing the change in this object by the change in ieee8021CnRpPortPriCreatedRps over that same time interval yields the average lifetime of an active RP during that interval.

"

REFERENCE

"802.1Qau clause 32.10.3, 32.13.1"

::= { ieee8021CnRpPortPriEntry 6 }

-- ****

-- The Reaction Point Group table. One table per end station.

-- One entry per Reaction Point.

-- ****

ieee8021CnRpGroupTable OBJECT-TYPE

SYNTAX SEQUENCE OF Ieee8021CnRpGroupEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Each row in this table supplies values for one or more Reaction Points (RPs). This table is indexed by component, port (interface), and an arbitrary RP index.

Creating an entry in ieee8021CnComptPriTable can create an entry in this table, with the default values, on all ports in the end station. An end station can have more than one RP for the same Congestion Notification Priority Value (CNPV), so creating an entry in ieee8021CnComptPriTable can create multiple entries in this table.

Because each port in a bridge component or end station can have a different relationship between CNPVs and RPs, the entries created or deleted on each port can be different.

Deleting the an entry in ieee8021CnCompntPriTable for a CNPV passing through the RP controlled by this entry deletes entries on some or all of the ports in the end station.

Because each port in an end station can have a different relationship between CNPVs and RPs, the entries created or deleted on each port can be different.

The relationship between ieee8021CnRpgIdentifier values and RPs is an implementation dependent matter.

The contents of this table SHALL be maintained across a restart of the system, except as noted in the description of ieee8021CnRpgIfIndex.

REFERENCE

"802.1Qau clause 12.21.6"

::= { ieee8021CnMIBObjects 8 }

```
ieee8021CnRpGroupEntry OBJECT-TYPE
    SYNTAX      Ieee8021CnRpGroupEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "An entry in the Reaction Point table controls a group of
         Reaction Points, on a port in an end station. All of the
         Reaction Point controlled by this entry serve the same
         Congestion Notification Priority Value.
    "
    INDEX { ieee8021CnRpgComponentId, ieee8021CnRpgPriority,
             ieee8021CnRpgIfIndex, ieee8021CnRpgIdentifier }
    ::= { ieee8021CnRpGroupTable 1 }
```

```
Ieee8021CnRpGroupEntry ::= SEQUENCE {
    ieee8021CnRpgComponentId      IEEE8021PbbComponentIdentifier,
    ieee8021CnRpgPriority        IEEE8021PriorityValue,
    ieee8021CnRpgIfIndex          InterfaceIndex,
    ieee8021CnRpgIdentifier      Unsigned32,
    ieee8021CnRpgEnable          TruthValue,
    ieee8021CnRpgTimeReset       TimeInterval,
    ieee8021CnRpgByteReset       Unsigned32,
    ieee8021CnRpgThreshold      Unsigned32,
    ieee8021CnRpgMaxRate         Unsigned32,
    ieee8021CnRpgAiRate          Unsigned32,
    ieee8021CnRpgHaiRate         Unsigned32,
    ieee8021CnRpgGd              Integer32,
    ieee8021CnRpgMinDecFac      Unsigned32,
    ieee8021CnRpgMinRate         Unsigned32
}
```

```
ieee8021CnRpgComponentId OBJECT-TYPE
    SYNTAX      IEEE8021PbbComponentIdentifier
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The bridge component within the system to which the information
         in this ieee8021CnRpGroupEntry 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.  
"  
 ::= { ieee8021CnRpGroupEntry 1 }
```

```
ieee8021CnRpgPriority OBJECT-TYPE  
SYNTAX      IEEE8021PriorityValue  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
"This object indicates the lowest numerical  
Congestion Notification Priority Value that this entry's  
Reaction Point serves.  
"  
REFERENCE  
"802.1Qau clause 12.21.5"  
 ::= { ieee8021CnRpGroupEntry 2 }
```

```
ieee8021CnRpgIfIndex OBJECT-TYPE  
SYNTAX      InterfaceIndex  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
"This object indicates the interface on which the group of  
Reaction Points reside.
```

Upon a restart of the system, the system SHALL, if necessary, change the value of this object, and rearrange the order of the ieee8021CnRpGroupTable, so that the value in ieee8021CnRpgIfIndex references the row in the ifXTable with the same value for ifAlias that it referenced before the system restart. If no such entry exists in the ifXTable, then the system SHALL delete the row in the ieee8021CnRpGroupTable.

```
"  
REFERENCE  
"802.1Qau clause 12.21.5"  
 ::= { ieee8021CnRpGroupEntry 3 }
```

```
ieee8021CnRpgIdentifier OBJECT-TYPE  
SYNTAX      Unsigned32 (1..4096)  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
"This object is an arbitrary integer indexing the entries in this table among the entries for the same interface. This index SHOULD, where possible, be equal to the Congestion Notification Priority Value served by this Reaction Point.  
"  
REFERENCE  
"802.1Qau clause 12.21.6"  
 ::= { ieee8021CnRpGroupEntry 4 }
```

```
ieee8021CnRpgEnable OBJECT-TYPE  
SYNTAX      TruthValue  
MAX-ACCESS  read-write  
STATUS      current  
DESCRIPTION  
"Controls the rpEnabled variable of the Reaction Point state machines of the Reaction Points (RPs) controlled by this
```

```

entry as follows:
    true(1)      The rpEnabled variable for the RPs controlled by
                  this object are not held in the FALSE state,
                  thus enabling them to pay attention to received
                  CNMs.
    false(2)     The rpEnabled variable for the RPs controlled by
                  this object are held in the FALSE state, thus
                  disabling them from paying attention to received
                  CNMs.
    "
REFERENCE
    "802.1Qau clause 32.11.1, 32.13.1"
DEFVAL { true }
 ::= { ieee8021CnRpGroupEntry 5 }

ieee8021CnRpgTimeReset OBJECT-TYPE
SYNTAX      TimeInterval
UNITS       "milliseconds"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the value for all of the state machine
     variables, rpgTimeReset, used to reset the timers RpWhile.
    "
REFERENCE
    "802.1Qau clause 32.11.2"
DEFVAL { 15 }
 ::= { ieee8021CnRpGroupEntry 6 }

ieee8021CnRpgByteReset OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "octets"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the value for all of the state machine
     variables, rpgByteReset, used to reset the counters
     rpByteCount.
    "
REFERENCE
    "802.1Qau clause 32.11.3"
DEFVAL { 150000 }
 ::= { ieee8021CnRpGroupEntry 7 }

ieee8021CnRpgThreshold OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the number of times rpByteStage or
     rpTimeStage can count before the
     RP rate control state machine advances states.
    "
REFERENCE
    "802.1Qau clause 32.11.4"
DEFVAL { 5 }
 ::= { ieee8021CnRpGroupEntry 8 }

ieee8021CnRpgMaxRate OBJECT-TYPE

```

```
SYNTAX      Unsigned32
UNITS       "Mbit/s"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the maximum rate, in multiples of 1 Mbit/s,
     at which an RP can transmit. Default value is the speed of the
     port. A system SHALL support a minimum value for this object
     that is no larger than 5 Mbits/s (object value 5). This rate
     includes all bits consequent to transmitting the frame on the
     LAN, including preamble, inter-frame gap, etc.
    "
REFERENCE
    "802.1Qau clause 32.11.5"
 ::= { ieee8021CnRpGroupEntry 9 }

ieee8021CnRpgAiRate OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "Mbit/s"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the transmission rate increment in the
     RPR_ACTIVE_INCREASE state (rpgAiRate) in multiples of 1 Mbit/s.
     This rate includes all bits consequent to transmitting the
     frame on the LAN, including preamble, inter-frame gap, etc.
    "
REFERENCE
    "802.1Qau clause 32.11.6"
DEFVAL { 5 }
 ::= { ieee8021CnRpGroupEntry 10 }

ieee8021CnRpgHaiRate OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "Mbit/s"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the transmission rate increment in the
     RPR_HYPER_INCREASE state (rpgHaiRate) in multiples of 1 Mbit/s.
     This rate includes all bits consequent to transmitting the
     frame on the LAN, including preamble, inter-frame gap, etc.
    "
REFERENCE
    "802.1Qau clause 32.11.7"
DEFVAL { 50 }
 ::= { ieee8021CnRpGroupEntry 11 }

ieee8021CnRpgGd OBJECT-TYPE
SYNTAX      Integer32
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
    "This object controls the number of bits that the value of the
     Quantized Feedback field received in a CNM PDU is shifted to
     the right to decrease rpTargetRate. rpgGd is thus 2 to the
     negative power of this object, e.g., 7 means rpgGd = 1/128.
    "
REFERENCE
```

```

    "802.1Qau clause 32.11.8"
DEFVAL { 7 }
 ::= { ieee8021CnRpGroupEntry 12 }

ieee8021CnRpgMinDecFac OBJECT-TYPE
SYNTAX      Unsigned32 (1..100)
UNITS       "percent"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"This object controls the minimum factor by which the current
RP transmit rate rpCurrentRate can be changed by reception
of a Congestion Notification Message. Its integer value
represents a percentage, from 1% to 100%.
"
REFERENCE
"802.1Qau clause 32.11.9"
DEFVAL { 50 }
 ::= { ieee8021CnRpGroupEntry 13 }

ieee8021CnRpgMinRate OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "Mbit/s"
MAX-ACCESS  read-write
STATUS      current
DESCRIPTION
"This object controls the minimum transmission rate (rpgMinRate)
in multiples of 1 Mbit/s. A system SHALL support a value for
this object that is no larger than 5 Mbit/s per second.
This rate includes all bits consequent to transmitting the
frame on the LAN, including preamble, inter-frame gap, etc.
"
REFERENCE
"802.1Qau clause 32.11.10"
DEFVAL { 5 }
 ::= { ieee8021CnRpGroupEntry 14 }

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

ieee8021CnCompliances OBJECT IDENTIFIER ::= { ieee8021CnConformance 1 }
ieee8021CnGroups      OBJECT IDENTIFIER ::= { ieee8021CnConformance 2 }

-- *****
-- Units of conformance
-- *****

ieee8021CnGlobalReqdGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnGlobalMasterEnable,
    ieee8021CnComPriLldpInstanceChoice,
    ieee8021CnComPriLldpInstanceSelector
}
STATUS      current
DESCRIPTION
"Objects in the global required group."

```

```
 ::= { ieee8021CnGroups 1 }

ieee8021CnCpGlobalGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnGlobalCnmTransmitPriority,
    ieee8021CnGlobalDiscardedFrames
}
STATUS      current
DESCRIPTION
    "Objects in the Congestion Point global group."
 ::= { ieee8021CnGroups 2 }

ieee8021CnCpidTranslateGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnCpidToIfComponentId,
    ieee8021CnCpidToIfIfIndex,
    ieee8021CnCpidToIfCpIndex
}
STATUS      current
DESCRIPTION
    "Objects in the CPID translate group."
 ::= { ieee8021CnGroups 3 }

ieee8021CnEplGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnEpIfIndex
}
STATUS      current
DESCRIPTION
    "Objects for the Errored Ports Table group."
 ::= { ieee8021CnGroups 4 }

ieee8021CnComPriGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnComPriDefModeChoice,
    ieee8021CnComPriAdminDefenseMode,
    ieee8021CnComPriCreation,
    ieee8021CnComPriRowStatus
}
STATUS      current
DESCRIPTION
    "Objects for the global per-priority group."
 ::= { ieee8021CnGroups 5 }

ieee8021CnCpPriGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnComPriAlternatePriority,
    ieee8021CnComPriAutoAltPri
}
STATUS      current
DESCRIPTION
    "Objects for the Congestion Point per-priority group."
 ::= { ieee8021CnGroups 6 }

ieee8021CnGlobalPriPortGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnPortPriDefModeChoice,
    ieee8021CnPortPriAdminDefenseMode,
    ieee8021CnPortPriAutoDefenseMode,
```

```

        ieee8021CnPortPriLldpInstanceChoice,
        ieee8021CnPortPriLldpInstanceSelector
    }
    STATUS      current
    DESCRIPTION
        "Objects for the global per-priority per-port group."
    ::= { ieee8021CnGroups 7 }

ieee8021CnCpPriPortGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnPortPriAlternatePriority
}
STATUS      current
DESCRIPTION
    "Objects for the Congestion Point per-priority per-port
     group.
    "
    ::= { ieee8021CnGroups 8 }

ieee8021CnCpGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnCpPriority,
    ieee8021CnCpMacAddress,
    ieee8021CnCpIdentifier,
    ieee8021CnCpQueueSizeSetPoint,
    ieee8021CnCpFeedbackWeight,
    ieee8021CnCpMinSampleBase,
    ieee8021CnCpDiscardedFrames,
    ieee8021CnCpTransmittedFrames,
    ieee8021CnCpTransmittedCnms,
    ieee8021CnCpMinHeaderOctets
}
STATUS      current
DESCRIPTION
    "Objects for the Congestion Point group."
    ::= { ieee8021CnGroups 9 }

ieee8021CnRpppGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnRpPortPriMaxRps,
    ieee8021CnRpPortPriCreatedRps,
    ieee8021CnRpPortPriCentiseconds
}
STATUS      current
DESCRIPTION
    "Objects for the Reaction Point per-Port per-priority group."
    ::= { ieee8021CnGroups 10 }

ieee8021CnRpGroup OBJECT-GROUP
OBJECTS {
    ieee8021CnRpgEnable,
    ieee8021CnRpgTimeReset,
    ieee8021CnRpgByteReset,
    ieee8021CnRpgThreshold,
    ieee8021CnRpgMaxRate,
    ieee8021CnRpgAiRate,
    ieee8021CnRpgHaiRate,
    ieee8021CnRpgGd,
    ieee8021CnRpgMinDecFac,
}

```

```
    ieee8021CnRpgMinRate
}
STATUS      current
DESCRIPTION
    "Objects for the Reaction Point group."
::= { ieee8021CnGroups 11 }

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

ieee8021CnBridgeCompliance MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for support by a bridge of
     the IEEE8021-CN-MIB module."

MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
MANDATORY-GROUPS {
    systemGroup
}

MODULE IF-MIB -- The interfaces MIB, RFC 2863
MANDATORY-GROUPS {
    ifGeneralInformationGroup
}

MODULE
MANDATORY-GROUPS {
    ieee8021CnGlobalReqdGroup,
    ieee8021CnCpGlobalGroup,
    ieee8021CnCpidTranslateGroup,
    ieee8021CnEplGroup,
    ieee8021CnComPriGroup,
    ieee8021CnCpPriGroup,
    ieee8021CnGlobalPriPortGroup,
    ieee8021CnCpPriPortGroup,
    ieee8021CnCpGroup
}

OBJECT ieee8021CnComPriRowStatus
SYNTAX      RowStatus { active(1), notInService(2),
                        notReady(3) }
WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                        destroy(6) }
DESCRIPTION "Support for createAndWait is not required."
::= { ieee8021CnCompliances 1 }

ieee8021CnStationCompliance MODULE-COMPLIANCE
STATUS      current
DESCRIPTION
    "The compliance statement for support by an end station of
     the IEEE8021-CN-MIB module."

MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
MANDATORY-GROUPS {
    systemGroup
```

```
}

MODULE IF-MIB -- The interfaces MIB, RFC 2863
MANDATORY-GROUPS {
    ifGeneralInformationGroup
}

MODULE
MANDATORY-GROUPS {
    ieee8021CnGlobalReqdGroup,
    ieee8021CnComPriGroup,
    ieee8021CnGlobalPriPortGroup,
    ieee8021CnRpppGroup,
    ieee8021CnRpGroup
}

GROUP ieee8021CnCpGlobalGroup
DESCRIPTION
"This group is optional and supports end stations that
choose to implement Congestion Points."

GROUP ieee8021CnCpidTranslateGroup
DESCRIPTION
"This group is optional and supports end stations that
choose to implement Congestion Points."

GROUP ieee8021CnEplGroup
DESCRIPTION
"This group is optional and supports end stations that
choose to implement Congestion Points."

GROUP ieee8021CnCpPriGroup
DESCRIPTION
"This group is optional and supports end stations that
choose to implement Congestion Points."

GROUP ieee8021CnCpPriPortGroup
DESCRIPTION
"This group is optional and supports end stations that
choose to implement Congestion Points."

GROUP ieee8021CnCpGroup
DESCRIPTION
"This group is optional and supports end stations that
choose to implement Congestion Points."


::= { ieee8021CnCompliances 2 }

END
```

17.7.14 Definitions of the IEEE8021-SRP MIB module

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

-- =====
-- MIB for support of 802.1Qat Stream Reservation Protocol
-- (SRP) in 802.1Q Bridges.
-- =====

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Counter64,
    Unsigned32
        FROM SNMPv2-SMI
    MacAddress,
    TEXTUAL-CONVENTION,
    TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    ieee802dot1mibs,
    IEEE8021PriorityCodePoint,
    IEEE8021VlanIndex
        FROM IEEE8021-TC-MIB
    IEEE8021FqtssTrafficClassValue
        FROM IEEE8021-FQTSS-MIB
    ieee8021BridgeBaseComponentId,
    ieee8021BridgeBaseEntry,
    ieee8021BridgeBasePort,
    ieee8021BridgeBasePortEntry
        FROM IEEE8021-BRIDGE-MIB
    BridgeId
        FROM BRIDGE-MIB
;

ieee8021SrpMib MODULE-IDENTITY
LAST-UPDATED "201102270000Z" -- February 27, 2011
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: Craig Gunther
    Postal: C/O IEEE 802.1 Working Group
            IEEE Standards Association
            445 Hoes Lane
            P.O. Box 1331
            Piscataway
            NJ 08855-1331
            USA
    E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"
DESCRIPTION
    "The Bridge MIB module for managing devices that support
the IEEE 802.1Q Stream Reservation Protocol."
```

Unless otherwise indicated, the references in this MIB module are to IEEE Std 802.1Q-2011.

Copyright (C) IEEE.
This version of this MIB module is part of IEEE802.1Q;
see the draft itself for full legal notices."
REVISION "201102270000Z" -- February 27, 2011
DESCRIPTION
"Minor edits to contact information etc. as part of
2011 revision of Std 802.1Q."

REVISION "201004190000Z" -- April 19, 2010
DESCRIPTION
"Initial revision, included in IEEE 802.1Qat"
 ::= { ieee802dot1mibs 19 }

-- =====
-- Textual Conventions
-- =====

IEEE8021SrpStreamRankValue ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"An 802.1 SRP Stream Rank value. This is an integer,
with the following interpretation placed on the value:

0: Emergency, high-rank stream,
1: Non-emergency stream."
REFERENCE "35.2.2.8.5b"
SYNTAX INTEGER {
 emergency(0),
 nonEmergency(1)
}

IEEE8021SrpStreamIdValue ::= TEXTUAL-CONVENTION
DISPLAY-HINT "1x:1x:1x:1x:1x:1x.1x:1x"
STATUS current
DESCRIPTION
"Represents an SRP Stream ID, which is often defined
as a MAC Address followed by a unique 16-bit ID."
SYNTAX OCTET STRING (SIZE (8))

IEEE8021SrpReservationDirectionValue ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"An 802.1 SRP Stream Reservation Direction value. This is
an integer, with the following interpretation placed on
the value:

0: Talker registrations,
1: Listener registrations."
REFERENCE "35.2.1.2"
SYNTAX INTEGER {
 talkerRegistrations(0),
 listenerRegistrations(1)
}

```
IEEE8021SrpReservationDeclarationTypeValue ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "An 802.1 SRP Stream Reservation Declaration Type value.
         This is an integer, with the following interpretation
         placed on the value:

        0: Talker Advertise,
        1: Talker Failed,
        2: Listener Asking Failed,
        3: Listener Ready,
        4: Listener Ready Failed."
    REFERENCE   "35.2.1.3"
    SYNTAX      INTEGER {
                    talkerAdvertise(0),
                    talkerFailed(1),
                    listenerAskingFailed(2),
                    listenerReady(3),
                    listenerReadyFailed(4)
                }
```

```
IEEE8021SrpReservationFailureCodeValue ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "An 802.1 SRP Stream Reservation Failure Code value.
         This is an integer, with the following interpretation
         placed on the value:

        0: No failure,
        1: Insufficient bandwidth,
        2: Insufficient Bridge resources,
        3: Insufficient bandwidth for Traffic Class,
        4: StreamID in use by another Talker,
        5: Stream destination address already in use,
        6: Stream pre-empted by higher rank,
        7: Reported latency has changed,
        8: Egress port is not AVBCapable,
        9: Use a different destination_address,
       10: Out of MSRP resources,
       11: Out of MMRP resources,
       12: Cannot store destination_address,
       13: Requested priority is not an SR Class priority,
       14: MaxFrameSize is too large for media,
       15: maxFanInPorts limit has been reached,
       16: Changes in FirstValue for a registered StreamID,
       17: VLAN is blocked on this egress port (Registration Forbidden),
       18: VLAN tagging is disabled on this egress port (untagged set),
       19: SR class priority mismatch."
    REFERENCE   "35.2.2.8.7"
    SYNTAX      INTEGER {
                    noFailure(0),
                    insufficientBandwidth(1),
                    insufficientResources(2),
                    insufficientTrafficClassBandwidth(3),
                    streamIDInUse(4),
                    streamDestinationAddressInUse(5),
                    streamPreemptedByHigherRank(6),
                    latencyHasChanged(7),
```

```

        egressPortNotAVBCapable(8),
        useDifferentDestinationAddress(9),
        outOfMSRPResources(10),
        outOfMMRPRResources(11),
        cannotStoreDestinationAddress(12),
        priorityIsNoAnSRClass(13),
        maxFrameSizeTooLarge(14),
        maxFanInPortsLimitReached(15),
        firstValueChangedForStreamID(16),
        vlanBlockedOnEgress(17),
        vlanTaggingDisabledOnEgress(18),
        srClassPriorityMismatch(19)
    }

-- =====
-- subtrees in the SRP MIB
-- =====

ieee8021SrpNotifications
OBJECT IDENTIFIER ::= { ieee8021SrpMib 0 }

ieee8021SrpObjects
OBJECT IDENTIFIER ::= { ieee8021SrpMib 1 }

ieee8021SrpConformance
OBJECT IDENTIFIER ::= { ieee8021SrpMib 2 }

ieee8021SrpConfiguration
OBJECT IDENTIFIER ::= { ieee8021SrpObjects 1 }

ieee8021SrpLatency
OBJECT IDENTIFIER ::= { ieee8021SrpObjects 2 }

ieee8021SrpStreams
OBJECT IDENTIFIER ::= { ieee8021SrpObjects 3 }

ieee8021SrpReservations
OBJECT IDENTIFIER ::= { ieee8021SrpObjects 4 }

-- =====
-- The ieee8021SrpConfiguration subtree
-- This subtree defines the objects necessary for the
-- operational management of SRP.
-- =====

ieee8021SrpBridgeBaseTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021SrpBridgeBaseEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table for SRP main control and status information.
All writeable objects in this table must be persistent
over power up restart/reboot. These objects augment
the ieee8021BridgeBasePortTable."
::= { ieee8021SrpConfiguration 1 }

ieee8021SrpBridgeBaseEntry OBJECT-TYPE

```

```
SYNTAX      Ieee8021SrpBridgeBaseEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "SRP control and status information for a bridge."
AUGMENTS { ieee8021BridgeBaseEntry }
 ::= { ieee8021SrpBridgeBaseTable 1 }

Ieee8021SrpBridgeBaseEntry ::=

SEQUENCE {
    ieee8021SrpBridgeBaseMsrpEnabledStatus
        TruthValue,
    ieee8021SrpBridgeBaseMsrpTalkerPruning
        TruthValue,
    ieee8021SrpBridgeBaseMsrpMaxFanInPorts
        Unsigned32,
    ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize
        Unsigned32
}

ieee8021SrpBridgeBaseMsrpEnabledStatus OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The administrative status requested by management for
     MSRP. The value true(1) indicates that MSRP should
     be enabled on this device, in all VLANs, on all ports
     for which it has not been specifically disabled. When
     false(2), MSRP is disabled, in all VLANs and on all
     ports, and all MSRP frames will be forwarded
     transparently. This object affects both Applicant and
     Registrar state machines. A transition from false(2)
     to true(1) will cause a reset of all MSRP state
     machines on all ports.

This object may be modified while the corresponding
instance of ieee8021BridgeBaseRowStatus is active(1).

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE  "35.2.1.4d"
DEFVAL     { true }
 ::= { ieee8021SrpBridgeBaseEntry 1 }

ieee8021SrpBridgeBaseMsrpTalkerPruning OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The value of the talkerPruning parameter which
     controls the propagation of Talker declarations.
     The value true(1) indicates that Talker attributes
     are only declared on ports that have the Stream
     destination_address registered in the MMRP MAC
     Address Registration Entries. When false(2),
     Talker attribute are declared on all egress ports
     in the active topology.
```

```

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE    "12.22.1, 35.2.1.4b, 35.2.4.3.1"
DEFVAL       { false }
 ::= { ieee8021SrpBridgeBaseEntry 2 }

ieee8021SrpBridgeBaseMsrpMaxFanInPorts OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The value of the msrpMaxFanInPorts parameter which
limits the total number of ports on a Bridge that
are allowed to establish reservations for inbound
Streams. A value of zero (0) indicates no fan-in
limit is being specified and calculations involving
fan-in will only be limited by the number of MSRP
enabled ports.

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE    "12.22.1, 35.2.1.4f"
DEFVAL       { 0 }
 ::= { ieee8021SrpBridgeBaseEntry 3 }

ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize OBJECT-TYPE
SYNTAX      Unsigned32
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
"The value of msrpLatencyMaxFrameSize parameter
which is used in the calculation of the maximum
latency through a bridge. The maximum size is
defined to be 2000 octets by default, but may be
set to a smaller or larger value dependent on the
particular Bridge configuration. This parameter
does not imply any type of policing of frame size,
it is only used in the latency calculations.

The value of this object MUST be retained across
reinitializations of the management system."
REFERENCE    "12.22.1, 35.2.1.4g"
DEFVAL       { 2000 }
 ::= { ieee8021SrpBridgeBaseEntry 4 }

ieee8021SrpBridgePortTable OBJECT-TYPE
SYNTAX      SEQUENCE OF Ieee8021SrpBridgePortEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
"A table for SRP control and status information about
every bridge port. Augments the ieee8021BridgeBasePortTable."
 ::= { ieee8021SrpConfiguration 2 }

ieee8021SrpBridgePortEntry OBJECT-TYPE
SYNTAX      Ieee8021SrpBridgePortEntry
MAX-ACCESS  not-accessible
STATUS      current

```

```
DESCRIPTION
    "SRP control and status information for a bridge port."
AUGMENTS { ieee8021BridgeBasePortEntry }
 ::= { ieee8021SrpBridgePortTable 1 }

Ieee8021SrpBridgePortEntry ::=

SEQUENCE {
    ieee8021SrpBridgePortMsrpEnabledStatus
        TruthValue,
    ieee8021SrpBridgePortMsrpFailedRegistrations
        Counter64,
    ieee8021SrpBridgePortMsrpLastPduOrigin
        MacAddress,
    ieee8021SrpBridgePortSrPvid
        IEEE8021VlanIndex
}

ieee8021SrpBridgePortMsrpEnabledStatus OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The administrative state of MSRP operation on this port. The
     value true(1) indicates that MSRP is enabled on this port
     in all VLANs as long as ieee8021BridgeMsrpEnabledStatus is
     also true(1). A value of false(2) indicates that MSRP is
     disabled on this port in all VLANs: any MSRP frames received
     will be silently discarded, and no MSRP registrations will be
     propagated from other ports. Setting this to a value of
     true(1) will be stored by the agent but will only take
     effect on the MSRP protocol operation if
     ieee8021BridgeMsrpEnabledStatus
     also indicates the value true(1). This object affects
     all MSRP Applicant and Registrar state machines on this
     port. A transition from false(2) to true(1) will
     cause a reset of all MSRP state machines on this port.

    The value of this object MUST be retained across
    reinitializations of the management system."
REFERENCE   "35.2.1.4e"
DEFVAL      { true }
 ::= { ieee8021SrpBridgePortEntry 1 }

ieee8021SrpBridgePortMsrpFailedRegistrations OBJECT-TYPE
SYNTAX      Counter64
UNITS       "failed MSRP registrations"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The total number of failed MSRP registrations, for any
     reason, in all VLANs, on this port.

    Discontinuities in the value of the counter can occur at
    re-initialization of the management system, and at other
    times as indicated by the value of ifCounterDiscontinuityTime
    object of the associated interface (if any)."
REFERENCE   "10.7.12.1"
 ::= { ieee8021SrpBridgePortEntry 2 }
```

```

ieee8021SrpBridgePortMsrpLastPduOrigin OBJECT-TYPE
    SYNTAX      MacAddress
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Source MAC Address of the last MSRP message
         received on this port."
    REFERENCE   "10.7.12.2"
    ::= { ieee8021SrpBridgePortEntry 3 }

ieee8021SrpBridgePortSrPvid OBJECT-TYPE
    SYNTAX      IEEE8021VlanIndex
    MAX-ACCESS  read-create
    STATUS      current
    DESCRIPTION
        "The default VLAN ID that Streams are assigned to.
         Talkers learn this VID from the SRP Domain attribute
         and tag Streams accordingly.

        The value of this object MUST be retained across
         reinitializations of the management system."
    REFERENCE   "35.2.2.8.3b"
    DEFVAL     { 2 }
    ::= { ieee8021SrpBridgePortEntry 4 }

-- =====
-- The ieee8021SrpLatency subtree
-- This subtree defines the objects necessary for retrieving
-- the latency of the various traffic classes on a port.
-- =====

-- =====
-- the ieee8021SrpLatencyTable
-- =====

ieee8021SrpLatencyTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpLatencyEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of latency measurement
         parameters for each traffic class."
    REFERENCE   "35.2.2.8.6"
    ::= { ieee8021SrpLatency 1 }

ieee8021SrpLatencyEntry OBJECT-TYPE
    SYNTAX      Ieee8021SrpLatencyEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing latency information
         for each traffic class. Rows in the table are
         automatically created for ports that are not an
         SRP domain boundary port (i.e. SRPdomainBoundaryPort
         is FALSE). Refer to Clause 6.6.4, 8.8.2, 12.21.3."
    INDEX    { ieee8021BridgeBaseComponentId,
              ieee8021BridgeBasePort,
              ieee8021SrpTrafficClass  }
    ::= { ieee8021SrpLatencyTable 1 }

```

```
Ieee8021SrpLatencyEntry ::=  
SEQUENCE {  
    ieee8021SrpTrafficClass  
        IEEE8021FqtssTrafficClassValue,  
    ieee8021SrpPortTcLatency  
        Unsigned32  
}  
  
ieee8021SrpTrafficClass OBJECT-TYPE  
SYNTAX      IEEE8021FqtssTrafficClassValue  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "The traffic class number associated with the  
row of the table.  
  
    Rows in the table are automatically created for  
ports that are not an SRP domain boundary port  
(i.e. SRPdomainBoundaryPort is FALSE)."  
REFERENCE   "6.6.4, 8.8.2, 12.21.3"  
::= { ieee8021SrpLatencyEntry 1 }  
  
ieee8021SrpPortTcLatency OBJECT-TYPE  
SYNTAX      Unsigned32  
UNITS       "nano-seconds"  
MAX-ACCESS  read-only  
STATUS      current  
DESCRIPTION  
    "The value of the portTcMaxLatency parameter for the  
traffic class. This value is expressed in  
nano-seconds."  
REFERENCE   "35.2.1.4, 35.2.2.8.6"  
::= { ieee8021SrpLatencyEntry 2 }  
  
-- =====  
-- The ieee8021SrpStreams subtree  
-- This subtree defines the objects necessary for retrieving  
-- the characteristics of the various Streams currently registered.  
-- =====  
  
-- =====  
-- the ieee8021SrpStreamTable  
-- =====  
ieee8021SrpStreamTable OBJECT-TYPE  
SYNTAX      SEQUENCE OF Ieee8021SrpStreamEntry  
MAX-ACCESS  not-accessible  
STATUS      current  
DESCRIPTION  
    "A table containing a set of characteristics  
for each registered Stream."  
REFERENCE   "35.2.2.8"  
::= { ieee8021SrpStreams 1 }  
  
ieee8021SrpStreamEntry OBJECT-TYPE  
SYNTAX      Ieee8021SrpStreamEntry  
MAX-ACCESS  not-accessible
```

```

STATUS      current
DESCRIPTION
    "A list of objects containing characteristics
    for each registered Stream. Rows in the table are
    automatically created for Streams registered on any
    port of a bridge."
INDEX  { ieee8021SrpStreamId }
::= { ieee8021SrpStreamTable 1 }

Ieee8021SrpStreamEntry ::==
SEQUENCE {
    ieee8021SrpStreamId
        IEEE8021SrpStreamIdValue,
    ieee8021SrpStreamDestinationAddress
        MacAddress,
    ieee8021SrpStreamVlanId
        IEEE8021VlanIndex,
    ieee8021SrpStreamTspecMaxFrameSize
        Unsigned32,
    ieee8021SrpStreamTspecMaxIntervalFrames
        Unsigned32,
    ieee8021SrpStreamDataFramePriority
        IEEE8021PriorityCodePoint,
    ieee8021SrpStreamRank
        IEEE8021SrpStreamRankValue
}
}

ieee8021SrpStreamId OBJECT-TYPE
SYNTAX      IEEE8021SrpStreamIdValue
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The Stream ID associated with the row of the table.

    Rows in the table are automatically created when
    Streams are registered via MSRP."
REFERENCE   "35.2.2.8.2"
::= { ieee8021SrpStreamEntry 1 }

ieee8021SrpStreamDestinationAddress OBJECT-TYPE
SYNTAX      MacAddress
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The MAC destination address for the Stream described
    by this reservation."
REFERENCE   "35.2.2.8.3a"
::= { ieee8021SrpStreamEntry 2 }

ieee8021SrpStreamVlanId OBJECT-TYPE
SYNTAX      IEEE8021VlanIndex
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The VLAN ID associated with the MSRP registration
    for this Stream."
REFERENCE   "35.2.2.8.3b"
::= { ieee8021SrpStreamEntry 3}

```

```
ieee8021SrpStreamTspecMaxFrameSize OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum size frame that will be sent by
         a Talker for this Stream. This value is part
         of the Traffic Specification for the Stream."
    REFERENCE  "35.2.2.8.4a"
    ::= { ieee8021SrpStreamEntry 4}

ieee8021SrpStreamTspecMaxIntervalFrames OBJECT-TYPE
    SYNTAX      Unsigned32 (0..65535)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum number of frame that will be sent
         during a class measurement interval (L.2). This
         value is part of the Traffic Specification for
         the Stream."
    REFERENCE  "35.2.2.8.4b, L.2"
    ::= { ieee8021SrpStreamEntry 5}

ieee8021SrpStreamDataFramePriority OBJECT-TYPE
    SYNTAX      IEEE8021PriorityCodePoint
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The Priority Code Point (PCP) value that the
         referenced Stream will be tagged with. This value
         is used to distinguish Class A and Class B traffic."
    REFERENCE  "35.2.2.8.5a"
    ::= { ieee8021SrpStreamEntry 6}

ieee8021SrpStreamRank OBJECT-TYPE
    SYNTAX      IEEE8021SrpStreamRankValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "SRP supports emergency and non-emergency.
         Emergency traffic will interrupt non-emergency
         traffic if there is insufficient bandwidth or
         resources available for the emergency traffic."
    REFERENCE  "35.2.2.8.5b"
    ::= { ieee8021SrpStreamEntry 7}

-- =====
-- The ieee8021SrpReservations subtree
-- This subtree defines the objects necessary for retrieving
-- the Stream attribute registrations on each port of a Bridge.
-- =====

-- =====
-- the ieee8021SrpReservationsTable
-- =====

ieee8021SrpReservationsTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021SrpReservationsEntry
    MAX-ACCESS  not-accessible
```

```

STATUS      current
DESCRIPTION
    "A table containing Stream attribute
    registrations per port."
REFERENCE   "35.2.4"
 ::= { ieee8021SrpReservations 1 }

ieee8021SrpReservationsEntry OBJECT-TYPE
SYNTAX      Ieee8021SrpReservationsEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A list of objects containing Stream attribute
    registrations per port. Rows in the table are
    automatically created for Streams registered on any
    port of a bridge."
INDEX   { ieee8021SrpReservationStreamId,
          ieee8021SrpReservationDirection,
          ieee8021BridgeBaseComponentId,
          ieee8021BridgeBasePort }
 ::= { ieee8021SrpReservationsTable 1 }

Ieee8021SrpReservationsEntry ::=

SEQUENCE {
    ieee8021SrpReservationStreamId
        IEEE8021SrpStreamIdValue,
    ieee8021SrpReservationDirection
        IEEE8021SrpReservationDirectionValue,
    ieee8021SrpReservationDeclarationType
        IEEE8021SrpReservationDeclarationTypeValue,
    ieee8021SrpReservationAccumulatedLatency
        Unsigned32,
    ieee8021SrpReservationFailureBridgeId
        BridgeId,
    ieee8021SrpReservationFailureCode
        IEEE8021SrpReservationFailureCodeValue,
    ieee8021SrpReservationDroppedStreamFrames
        Counter64,
    ieee8021SrpReservationStreamAge
        Unsigned32
}

ieee8021SrpReservationStreamId OBJECT-TYPE
SYNTAX      IEEE8021SrpStreamIdValue
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The Stream ID associated with the row of the table.

    Rows in the table are automatically created when
    Streams are registered via MSRP."
REFERENCE   "35.2.2.8.2"
 ::= { ieee8021SrpReservationsEntry 1 }

ieee8021SrpReservationDirection OBJECT-TYPE
SYNTAX      IEEE8021SrpReservationDirectionValue
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION

```

```
        "The source of this Stream registration, either
        Talker or Listener."
REFERENCE    "35.2.1.2"
 ::= { ieee8021SrpReservationsEntry 2 }

ieee8021SrpReservationDeclarationType OBJECT-TYPE
SYNTAX      IEEE8021SrpReservationDeclarationTypeValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "The type of Talker or Listener registration."
REFERENCE    "35.2.1.3"
 ::= { ieee8021SrpReservationsEntry 3 }

ieee8021SrpReservationAccumulatedLatency OBJECT-TYPE
SYNTAX      Unsigned32
UNITS       "nano-seconds"
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "The Accumulated Latency associated with the current
registration.

For Talker registrations this represents the accumulated
latency from the Talker to the ingress port of this
Bridge.

For Listener registrations this represents the accumulated
latency to the ingress port of the neighbor Bridge or
end stations. This include the latency of the media
attached to this egress port."
REFERENCE    "35.2.2.8.6"
 ::= { ieee8021SrpReservationsEntry 4 }

ieee8021SrpReservationFailureBridgeId OBJECT-TYPE
SYNTAX      BridgeId
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "The first Bridge that changes a Talker Advertise to a
Talker Failed registration will report its Bridge
Identification in this field. That single Bridge
Identification is then propagated from Bridge to Bridge."
REFERENCE    "35.2.2.8.7a"
 ::= { ieee8021SrpReservationsEntry 5 }

ieee8021SrpReservationFailureCode OBJECT-TYPE
SYNTAX      IEEE8021SrpReservationFailureCodeValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
        "The first Bridge that changes a Talker Advertise to a
Talker Failed registration will report the Failure Code
in this field. That single Failure Code is then propagated
from Bridge to Bridge."
REFERENCE    "35.2.2.8.7b"
 ::= { ieee8021SrpReservationsEntry 6 }

ieee8021SrpReservationDroppedStreamFrames OBJECT-TYPE
```

```

SYNTAX      Counter64
UNITS      "frames"
MAX-ACCESS  read-only
STATUS     current
DESCRIPTION
    "A count of the number of data stream frames that have
    been dropped for whatever reason. These are not MSRP
    frames, but the stream data frames that are carried by
    the MSRP Reservation.

Discontinuities in the value of the counter can occur at
re-initialization of the management system, and at other
times as indicated by the value of ifCounterDiscontinuityTime
object of the associated interface (if any)."
REFERENCE   "35.2.5.1"
 ::= { ieee8021SrpReservationsEntry 7 }

ieee8021SrpReservationStreamAge OBJECT-TYPE
SYNTAX      Unsigned32
UNITS      "seconds"
MAX-ACCESS  read-only
STATUS     current
DESCRIPTION
    "The number of seconds since the reservation was established
    on this port. Talkers shall report this as the seconds
    since the first receipt of the Talker Advertise or Talker
    Failed. Listeners shall report this as the number of
    seconds since the destination_address was first added to
    the Dynamic Reservations Entries."
REFERENCE   "35.2.1.4c"
 ::= { ieee8021SrpReservationsEntry 8 }

-- =====
-- IEEE8021 SRP MIB - Conformance Information
-- =====

ieee8021SrpCompliances
OBJECT IDENTIFIER ::= { ieee8021SrpConformance 1 }
ieee8021SrpGroups
OBJECT IDENTIFIER ::= { ieee8021SrpConformance 2 }

-- =====
-- units of conformance
-- =====

-- =====
-- the ieee8021SrpConfiguration group
-- =====

ieee8021SrpConfigurationGroup OBJECT-GROUP
OBJECTS {
    ieee8021SrpBridgeBaseMsrpEnabledStatus,
    ieee8021SrpBridgeBaseMsrpTalkerPruning,
    ieee8021SrpBridgeBaseMsrpMaxFanInPorts,
    ieee8021SrpBridgeBaseMsrpLatencyMaxFrameSize,
    ieee8021SrpBridgePortMsrpEnabledStatus,
    ieee8021SrpBridgePortMsrpFailedRegistrations,
    ieee8021SrpBridgePortMsrpLastPduOrigin,
}

```

```
        ieee8021SrpBridgePortSrPvid
    }
    STATUS      current
    DESCRIPTION
        "Objects that define configuration of SRP."
    ::= { ieee8021SrpGroups 1 }

-- =====
-- the ieee8021SrpLatency group
-- =====

ieee8021SrpLatencyGroup OBJECT-GROUP
    OBJECTS {
        ieee8021SrpPortTcLatency
    }
    STATUS      current
    DESCRIPTION
        "Objects that define latency for SRP."
    ::= { ieee8021SrpGroups 2 }

-- =====
-- the ieee8021SrpStreams group
-- =====

ieee8021SrpStreamsGroup OBJECT-GROUP
    OBJECTS {
        -- ieee8021SrpStreamId,
        ieee8021SrpStreamDestinationAddress,
        ieee8021SrpStreamVlanId,
        ieee8021SrpStreamTspecMaxFrameSize,
        ieee8021SrpStreamTspecMaxIntervalFrames,
        ieee8021SrpStreamDataFramePriority,
        ieee8021SrpStreamRank
    }
    STATUS      current
    DESCRIPTION
        "Objects that define Streams for SRP."
    ::= { ieee8021SrpGroups 3 }

-- =====
-- the ieee8021SrpReservations group
-- =====

ieee8021SrpReservationsGroup OBJECT-GROUP
    OBJECTS {
        -- ieee8021SrpReservationStreamId,
        -- ieee8021SrpReservationDirection,
        ieee8021SrpReservationDeclarationType,
        ieee8021SrpReservationAccumulatedLatency,
        ieee8021SrpReservationFailureBridgeId,
        ieee8021SrpReservationFailureCode,
        ieee8021SrpReservationDroppedStreamFrames,
        ieee8021SrpReservationStreamAge
    }
    STATUS      current
    DESCRIPTION
        "Objects that define Stream Reservations for SRP."
    ::= { ieee8021SrpGroups 4 }
```

```
-- =====  
-- compliance statements  
-- =====  
  
ieee8021SrpCompliance MODULE-COMPLIANCE  
    STATUS      current  
    DESCRIPTION  
        "The compliance statement for devices supporting  
        Stream Reservation Protocol.  
  
        Support of the objects defined in the IEEE8021-SRP MIB  
        also requires support of the IEEE8021-BRIDGE-MIB; the  
        provisions of 17.3.2 apply to implementations claiming  
        support of the IEEE8021-SRP MIB."  
  
MODULE -- this module  
MANDATORY-GROUPS {  
    ieee8021SrpConfigurationGroup,  
    ieee8021SrpLatencyGroup,  
    ieee8021SrpStreamsGroup,  
    ieee8021SrpReservationsGroup  
}  
  
 ::= { ieee8021SrpCompliances 1 }  
  
END
```

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, 6.17) 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 transmitted 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 Continuity 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 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, 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. A single fault may be detected by multiple MEPs and may result in the generation of multiple Fault Alarms. In the absence of alarm suppression mechanisms in CFM, network management systems receiving such Fault Alarms can deploy other means to identify if a Fault Alarm identifies the primary fault for which a fault corrective action can be initiated, or represents a secondary fault for which no corrective action can be taken.

For VLANs, fault recovery is provided by the active topology protocols of Clause 13 while fault recovery for TESIs is provided by the Protection Switching mechanisms of 26.10. Fault recovery actions performed by the network administrator, such as the correction of configuration errors or replacement of failed components, 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 EIIS 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.

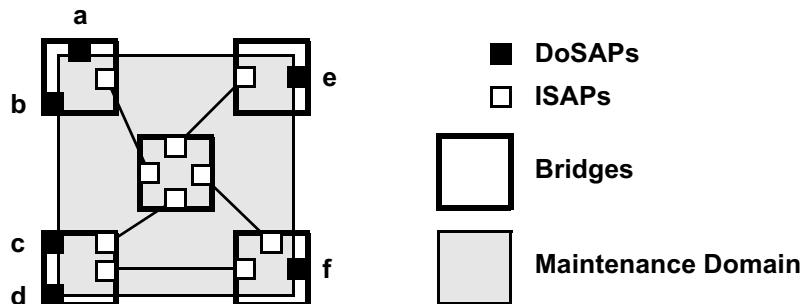


Figure 18-1—One Maintenance Domain: operator's view

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 (13.5.3), interconnected via the CIST (13.5.2), 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 (02/2008), 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.

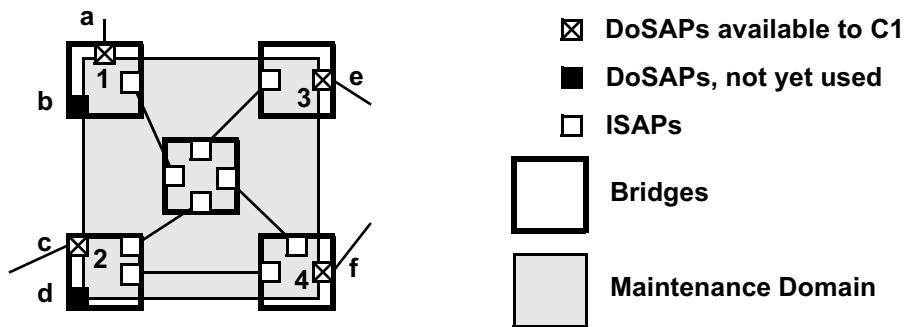


Figure 18-2—One service instance: operator’s view

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.

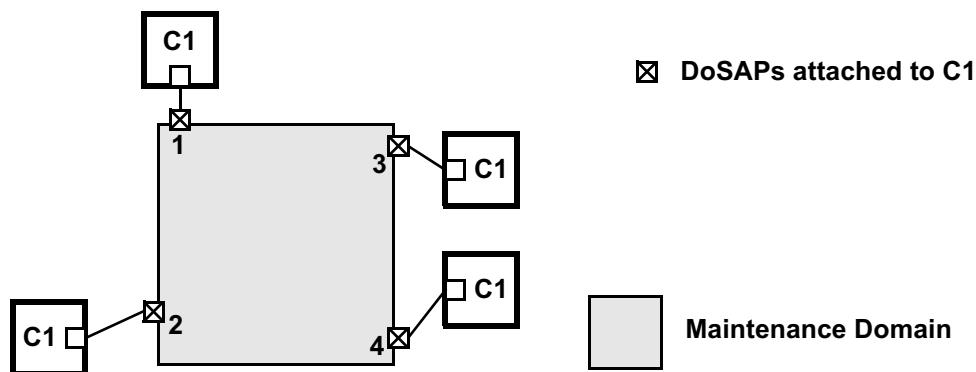


Figure 18-3—One service instance: customer’s view

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.

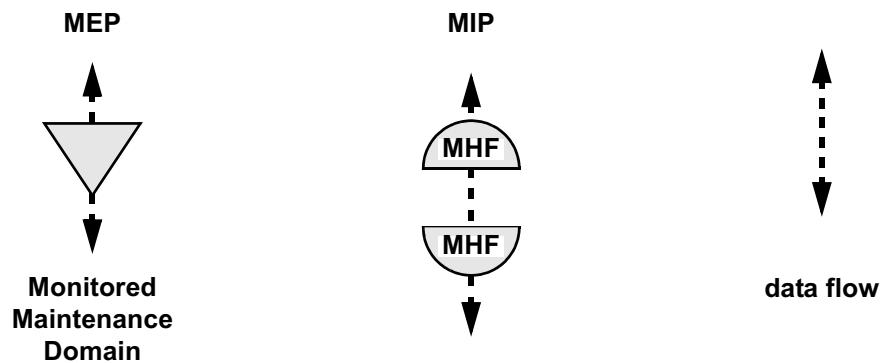


Figure 18-4—MEP and MIP Symbols

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:

- 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.
- 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.
- 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.
- 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.
 - 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 nonstandard configuration from the operator clouds on either side, a network of provider backbone Bridges, a network of Ethernet over SONET circuits with a central interconnect device, or an ATM Emulated LAN.²⁹

- 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.

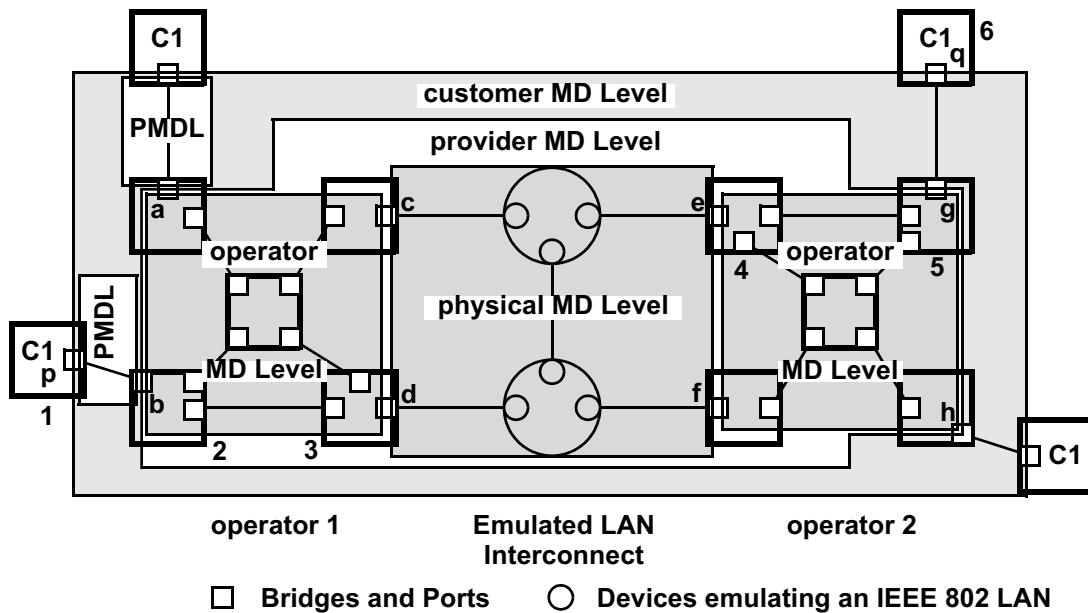


Figure 18-5—Maintenance Associations: one service instance in a provider network

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

²⁹See ATM Forum af-lane-0021.000; “LAN Emulation over ATM 1.0”; http://www.mtaforum.org/tech/atm_specs.shtml.

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.

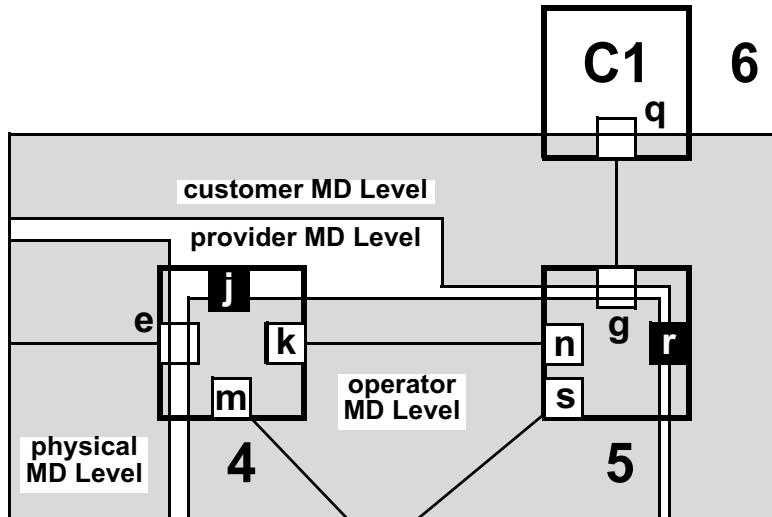


Figure 18-6—Maintenance Associations: Expansion of Figure 18-5

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.

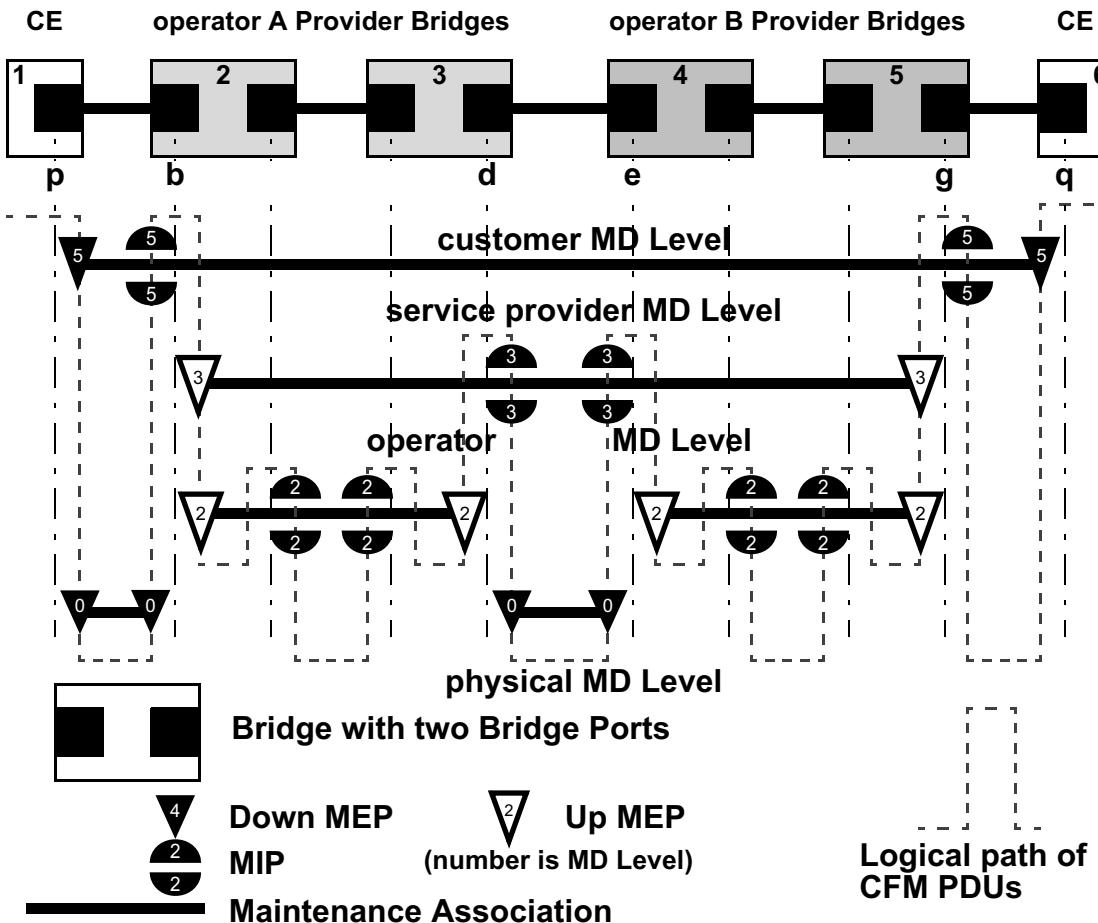


Figure 18-7—MEPs, MIPs, and MD Levels

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.

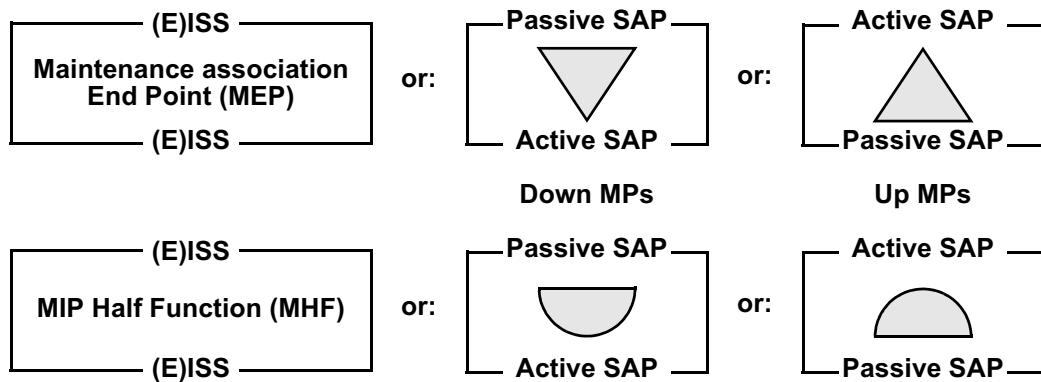


Figure 19-1—CFM Protocol shims

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; or
- d) An I-SID in the case of a backbone service instance MA; or
- e) A set of 3-tuples <MAC-ESP DA, MAC-ESP SA, ESP-VID> inherited from the MA.

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

- f) A Maintenance Domain Level (MD Level), inherited from its Maintenance Domain; and
- g) A Primary VID, or an I-SID in the case of a Backbone Service Instance, or a set of CBP MAC addresses and a set of ESP-VIDs in the case of a TESI, 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:

- h) The Bridge Port on which the MEP is configured; and
- i) 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 or I-SID or series of <ESP-DA, ESP-SA, ESP-VID> 3-tuples 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 20.51;
- 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;
- j) Maintains the variables in 20.9;
- k) May decapsulate and transmit the payload of received Send Frame Messages (SFMs) (29.2.6);
- l) May pass the received RFMs to RFM-Receiver (29.2.4).

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

- m) 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.
- n) 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.
- o) The `canonical_format_indicator` emitted on any frame is the appropriate value corresponding to the MAC attached to the Bridge Port.
- p) 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. 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 MFH. 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.

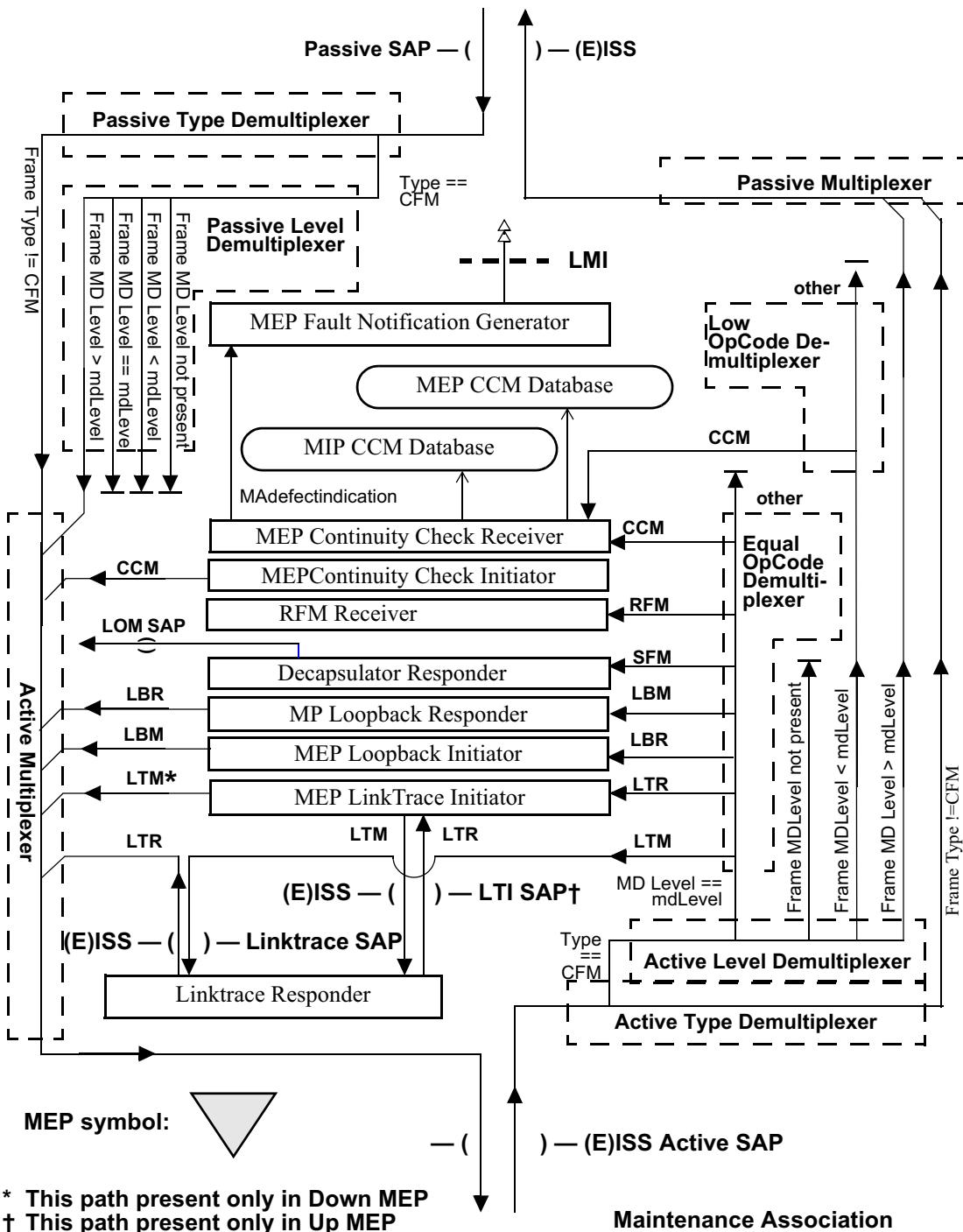


Figure 19-2—Maintenance association End Point (MEP)

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 6.5.3 (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 (02/2008) 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.

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.

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, to update the MEP CCM Database. The MEP Continuity Check Receiver maintains one instance of the Remote MEP state machine (20.20) for each active remote 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 or I-SID 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.

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 MHFrecvCCM and MHFCCMPDU for 20.15 and transfers CCM to Passive MHF Multiplexer
LBM	Sets LBMreceived and LBMPDU for 20.29	Discards PDU	Sets LBMreceived and LBMPDU for 20.29 and in non-PBB-TE MHFs transfers LBM to Passive MHF Multiplexer
LBR	Sets LBRreceived and LBRPDU for 20.34	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 20.45	Discards PDU	Transfers LTR to Passive MHF Multiplexer
Other	Discards PDU	Discards PDU	Transfers PDU to Passive MHF Multiplexer
SFM	Sets SFMreceived (29.3.8.5) and SFMPDU (29.3.8.3)	Discards PDU	Sets SFMreceived (29.3.8.5) and SFMPDU (29.3.8.3). Transfers the SFM to Passive MHF Multiplexer
RFM	Sets RFMreceived (29.3.6.1) and RFMPDU (29.3.6.2)	Discards PDU	Sets RFMreceived (29.3.6.1) and RFMPDU (29.3.6.2). Transfers the RFM to Passive MHF Multiplexer

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 the PBB-TE MIP TLV (21.7.5) in the case of a PBB-TE MA and transmits LBRs in response. The state machine for the MP Loopback Responder is described in 20.29.

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 20.32.

19.2.12 MEP Linktrace Initiator

The Linktrace Initiator transmits LTM 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 20.45. 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 toward 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 20.37.

19.2.17 MEP Decapsulator Responder

The MEP Decapsulator Responder is an optional component entity within an MEP shim required only by DDCFM. Its position relative to other MP component entities is shown in Figure 19-2. When the destination_address of the Send Frame Message (SFM) matches with the MEP's MAC address, the MEP Decapsulator Responder (29.2.6) will decapsulate the SFM, change the source_address field of the decapsulated data frame if it is required, and send the frame to the LOM SAP of selected port(s) (29.3.9.1).

19.2.18 MEP RFM Receiver

The MEP RFM Receiver is an optional component entity within a MEP shim required only by DDCFM. It is placed next to the OpCode Demultiplexer functions receiving RFM frames. When the destination_address of the Reflected Frame Message (RFM) matches with the MPs MAC address, the MEP RFM Receiver will pass the received RFM to the corresponding, nonstandardized RFM Analyzer.

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; or a set of I-SIDs, one for 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 20.51;
- b) Optionally, validates received CCMs and builds the MIP CCM Database;
- c) Responds to LBMs with LBRs as defined in 20.2.2;
- d) Responds to LTMs by forwarding them and responding with LTRs as defined in 20.3.2;
- e) May decapsulate and transmit the payload of the received Send Frame Message (29.4.3);
- f) May pass the received Reflected Frame Message (29.4.2).

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

- g) 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.
- h) All LBMs forwarded by the MHF carry the same `vlan_identifier`, `canonical_format_indicator`, and `drop_eligible` parameters as the received LBM.
- i) 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, or to the Reverse VID field of the PBB-TE MIP TLV if the received LBM or LTM carries such a TLV.

- j) The canonical_format_indicator emitted on any frame is the appropriate value corresponding to the MAC attached to the Bridge Port.
- k) 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.

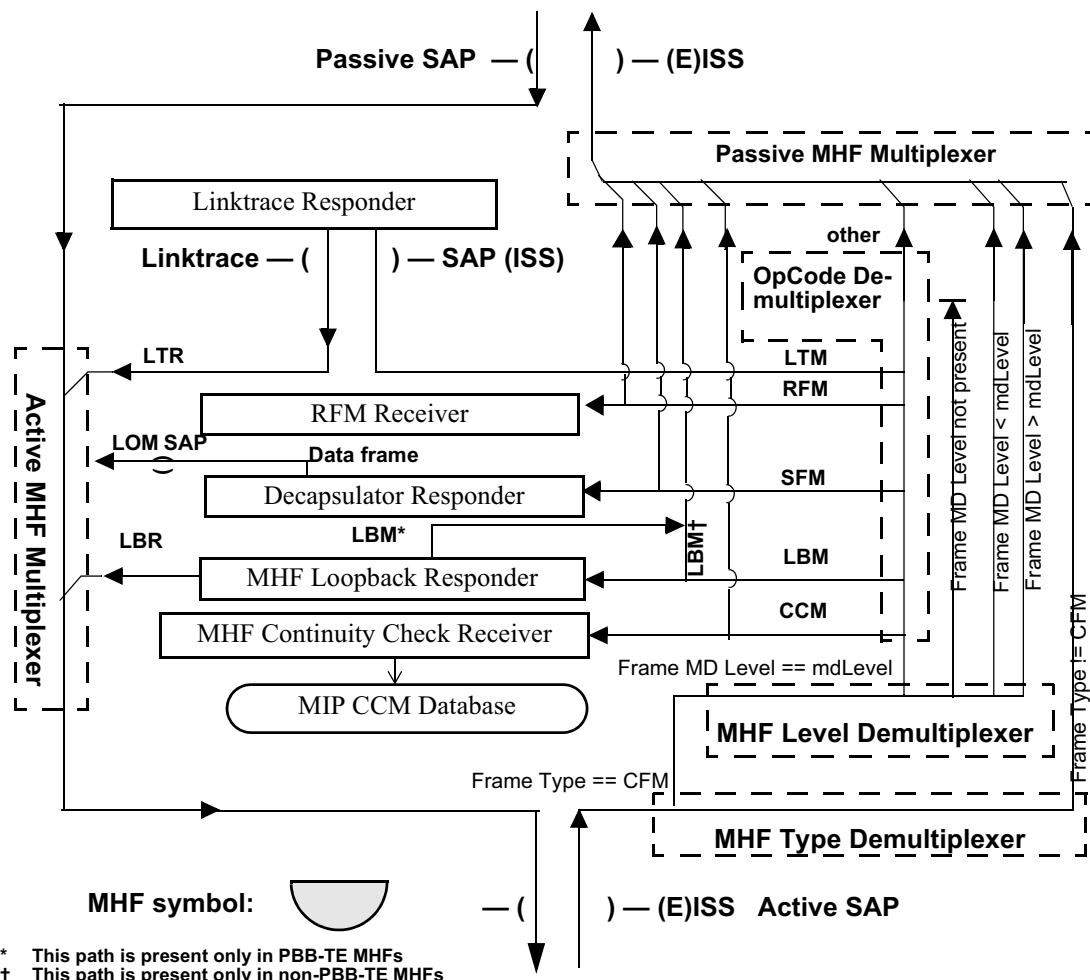


Figure 19-3—MIP Half Function

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.

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 LTM to the Linktrace Responder, and to pass LTRs from Linktrace Responder to the MHF’s Active SAP.

19.3.12 MHF Decapsulator Responder

The MHF Decapsulator Responder is identical to the MEP Decapsulator Responder described in 19.2.17.

19.3.13 MHF RFM Receiver

The MHF RFM Receiver is identical to the MEP RFM Receiver described in 19.2.18.

19.4 Maintenance Point addressing

The CFM entities within an MP use the Group addresses for CCM and LTM PDUs listed in Table 8-13 and Table 8-14 and in the case of PBB-TE MAs, the Individual MAC Addresses or the Group MAC addresses that are associated with the monitored TESI (20.1, 20.2, 20.3). In addition, they recognize and in the case of PBB-TE MEPs use the Individual MAC address of the port on which the MP is operating. This is the 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). CFM entities associated with a TESI may also use the ESP-SA parameter in the tuple identifying the corresponding ESP.

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 the Individual MAC address of the MP or are a group address. Each of these entities (except the MEP Continuity Check Receiver) filters and does not process frames whose destination_address parameters are not a group address and do not match the Individual MAC address of the MP.

NOTE—Even though in the original version of IEEE Std 802.1ag-2007, the MEP Continuity Check Initiator could only transmit multicast CCMs, the MEP Continuity Check Receiver recognized unicast CCMs. In general the definitions of the various CFM receiving entities were more liberal in recognizing certain Individual or Group MAC addresses than the corresponding definitions of the CFM transmitting entities a property that allows for the PBB-TE enhancements of CFM, and for improved compatibility with ITU-T Y.1731 (02/2008).

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.

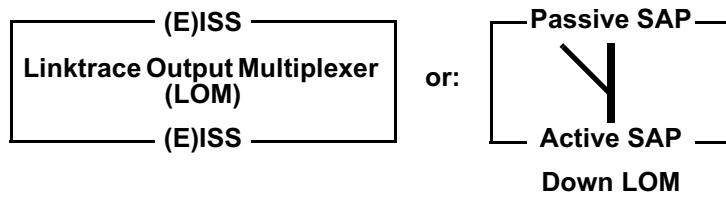


Figure 19-4—Linktrace Output Multiplexer shim

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.

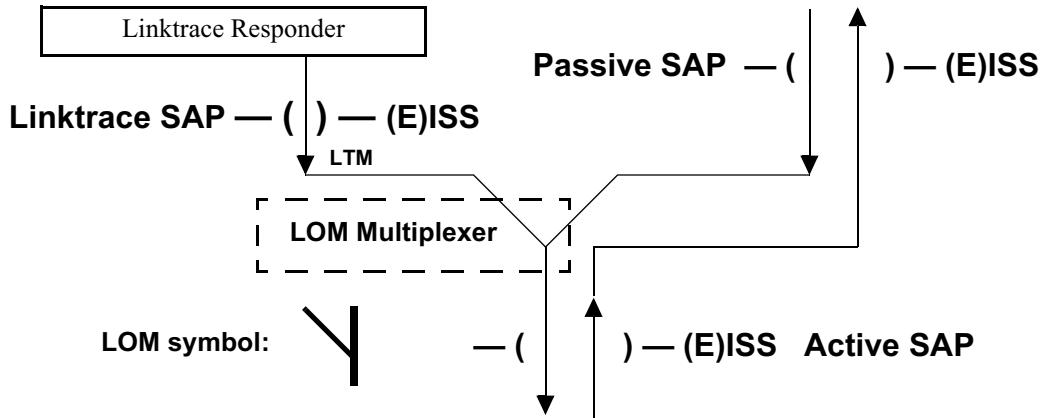


Figure 19-5—Linktrace Output Multiplexer architecture

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 toward 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 toward 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 VID, 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, toward the LAN, and not toward 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 20-16 for a diagrammatic explanation of these paths.

Table 19-2 summarizes the SAP connections between the MPs and LOM on the one hand, and the Linktrace Responder, on the other.

Table 19-2—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 20.50.

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);
- c) Linktrace protocol (20.3); and
- d) DDCFM protocol (29.3).

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, 13.22, and Figure 13-14.

This clause specifies the following:

- e) Relationships among the various CFM state machines and their variables (20.4);
- f) Requirements for decrementing the timer counters used by a number of CFM state machines (20.5);
- g) Maintenance Domain variables (20.7) that control all Maintenance Associations belonging to the Maintenance Domain;
- h) Maintenance Association variables (20.8) that control all MEPs and MHFs belonging to the Maintenance Association;
- i) 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;
- j) MEP Continuity Check Initiator (19.2.9) variables (20.10), procedures (20.11), and state machine (20.12);
- k) (optional) MHF Continuity Check Receiver (19.3.9) variables (20.13), procedures (20.14), and state machine (20.15);
- l) MEP Continuity Check Receiver (19.2.8) variables (20.16), procedures (20.17), and state machine (20.18);
- m) Remote MEP variables (20.19) and state machines (20.20) used by the MEP Continuity Check Receiver to track each active Remote MEP [item ae] in 12.14.7.1.3];
- n) 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;
- o) 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;
- p) MP Loopback Responder (19.2.10) variables, (20.27), procedures (20.28), and state machine (20.29);
- q) MEP Loopback Initiator (19.2.11) variables (20.30), transmit procedures (20.31) and state machine (20.32), and receive procedures (20.33) and state machine (20.34);
- r) MEP Fault Notification Generator (19.2.16) variables (20.35), procedures (20.36), and state machine (20.37);
- s) MEP Linktrace Initiator (19.2.12) variables (20.41) and transmit procedures (20.42), and receive variables (20.43), procedures (20.44), and state machine (20.45);
- t) Linktrace Responder variables (20.46), procedures (20.47), and state machine (20.48), used by the Linktrace Responder (19.6); and
- u) LTR Transmitter state machine (20.50) used only by the Linktrace Responder.

To facilitate extension of this standard while maximizing interoperability, this clause further specifies:

- v) Rules for CFM PDU validation and versioning (20.51) that support specific goals for extensibility and interoperability (20.51.1);

- w) Rules for the identification of incoming CFM PDUs (20.52); and
- x) Rules for the use of transaction identifier and sequence number fields (20.53).

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 one of the following:

- 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;
- 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; or
- f) Reception by a PBB-TE MEP implementing the Traffic field, over a period of time that is 4.75 times the CCM interval or equal to 50 ms, whichever is larger, of CCMs with Traffic fields not matching the local Traffic field declaration, indicating a mismatch on the presence or absence of backbone service instance entries on the CBPs that are associated with the monitored TESI.

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.

Depending on the associated 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 8-13 or, in the case of a PBB-TE associated MA, in a unicast or multicast frame with a destination_address identified by the monitored MA. 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.

NOTE—The TESI Multiplex Entity (6.19) helps PBB-TE associated MEPs to avoid this type of cross connect problem for unicast CCMs by checking the source_address to determine CCMs destined to PBB-TE MAs. Correspondingly the assignment of globally unique MAIDs is not as important for PBB-TE MAs as it is for other types of MAs but in general a globally unique MAID will provide guarantees for full coverage on all possible misconfiguration errors.

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 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 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.³⁰ 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 (20.37), or the MEP Mismatch Fault Notification Generator state machine (20.40), detects that a configured time period (default, 2.5 s) has passed with one or more defects indicated, and Fault Alarms are enabled. Each of these 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 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.

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. The mmddefectIndication (20.38.2) is independent from the other CFM defects and is reported independently by the operation of the MEP Mismatch Fault Notification Generator state machine (20.40).

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. This check is optional for CCMs associated with PBB-TE MAs. 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.

³⁰See, for example, ITU-T G.806 (2006) [B33].

Table 20-1—Fault Alarm defects and priorities

Defect		Priority	
Variable	highestDefect (20.35.9)	highestDefectPri (20.35.8)	Importance
xconCCMdefect (20.23.3)	DefXconCCM	5	Most
errorCCMdefect (20.21.3)	DefErrorCCM	4	
someRMEPCCMdefect (20.35.5)	DefRemoteCCM	3	
someMACstatusDefect (20.35.6)	DefMACstatus	2	
someRDIDefect (20.35.7)	DefRDICCM	1	Least

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 8.8.8 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

The Loopback protocol 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. In the case of a PBB-TE MA, the destination_address parameter for an LBM is always a MAC address associated with the terminating CBP(s). In order to identify intermediate MHFs in the PBB-TE MA, an additional TLV, the PBB-TE MIP TLV (21.7.5), is carried as the first TLV in an LBM, with the receiving MHF's MAC address in its MIP MAC address field (21.7.5.1). The receiving MP responds to the LBM with a 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. In a PBB-TE MA, a PBB-TE MIP TLV is carried by the transmitted LBM as the first TLV if the targeted MP is a MIP. The MIP MAC address field (21.7.5.1) in the TLV is set to the destination_address parameter of the appropriate managed object [item b) in 12.14.7.3.2]. The PBB-TE MIP TLV provides also the Reverse VID field [21.7.5.2, item f) in 12.14.7.3.2] to be used in the VID field by a MIP transmitted LBR, and, when the LBM is to be sent on a

point-to-multipoint TESI, the Reverse MAC field (21.7.5.3) associated with MAC addresses to be used by the LBR. 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 does not match the MAC address of the receiving MP, and the MP Loopback Responder does not reside in a PBB-TE MHF, the MP shall discard the LBM. If the receiving MP Loopback Responder resides in a non-PBB-TE associated MHF and the destination_address is a Group MAC address, the MHF shall discard the LBM. If the MP Loopback Responder state machine resides in a PBB-TE MEP and the received LBM carries a PBB-TE MIP TLV, the MEP shall discard the LBM. PBB-TE MHFs, forward all received LBMs and only stop and process those that carry a PBB-TE MIP TLV containing in their MIP MAC address field the MAC address of the receiving PBB-TE MIP [item f) in 20.28.1]. 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. In the case of a PBB-TE MHF this destination_address value will be overwritten by the value of the Group MAC address in the Reverse MAC field of the PBB-TE MIP TLV, if this is present. In the case of a PBB-TE MEP, the destination_address parameter for the transmitted LBR is the value of the ESP-DA of the MA's ESP that has the replying MEP's MAC address [item i) in 12.14.7.1.3] in its ESP-SA field;
- b) The source_address parameter for the LBR is the MAC address of the replying MP. If the MP is associated with a PBB-TE MHF, the source_address is set to the destination_address of the received LBM if this is an Individual MAC address or otherwise to the value carried in the Reverse MAC field, contained in the PBB-TE MIP TLV;
- c) The OpCode field is changed from LBM to LBR; and
- d) (Only in PBB-TE MAs) The vlan_identifier for the LBR is the Primary VID [item d) in 12.14.7.1.3] associated with the replying PBB-TE MEP; otherwise if the received LBM contained a PBB-TE MIP TLV, the value in the Reverse VID field is used as the vlan_identifier parameter.

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 20.51, 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 (02/2008) 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 20.52, 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. It is carried in a frame, with a destination_address taken from Table 8-14 according to the MD Level of the transmitting MEP, or the ESP-DA field of the appropriate ESP in the case of a PBB-TE MA (20.31.1), 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 and vlan_identifier to an 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, or if the MP is associated with a PBB-TE MA, 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 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

LTM 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, 20.41.1] 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 value³¹ 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 (20.41.2). 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.

In the case of a VID-based MA, 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 8-14. In the case of a PBB-TE MEP the destination_address of an LTM is the entry in the ESP-DA field of the 3-tuple of the MA’s ESP that has the MEP’s MAC address in its ESP-SA field.

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.

In the case of a PBB-TE MA, a PBB-TE MIP TLV is carried by the LTM, containing the Reverse VID [21.7.5.2, item e) in 12.14.7.4.2], to be used in the VID field by a MIP transmitted LTR, and, when the LTM is to be sent on a point-to-multipoint TESI, the Reverse MAC field (21.7.5.3) associated with MAC addresses to be used by the corresponding LTRs. The MIP MAC address field (21.7.5.1) in a PBB-TE MIP TLV associated with an LTM is null.

The MEP Linktrace Initiator in a Down MEP transmits the LTM through its Active SAP, toward 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 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 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 20.51.4; invalid or incorrectly addressed LTMs are discarded without further processing. After processing the LTM, the Linktrace Responder can return an LTR

³¹For this purpose, a pseudo-random value is sufficient, provided that the same value is not produced each time a system is reinitialized.

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

- 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 nonzero 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 via an MHF Linktrace SAP or a MEP LTI SAP, and not a MEP Linktrace SAP;
- 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.

The `source_address` parameter in the forwarded LTM does not change if the MA is associated to a TESI.

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. However, in the case of TESIs, the forwarded LTM simply uses the source address of the original LTM since the previous considerations do not apply in a PBB-TE Region. 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 20.51, but are not specified in this standard, and it ignores any valid Organization-Specific TLV not known to it. Except as otherwise specified in 20.47.3 and 20.47.4, 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 (20.41.2), 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 13.22 and of Figure 13-13: 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 MFH and

Linktrace Responder state machines, because they operate independently.

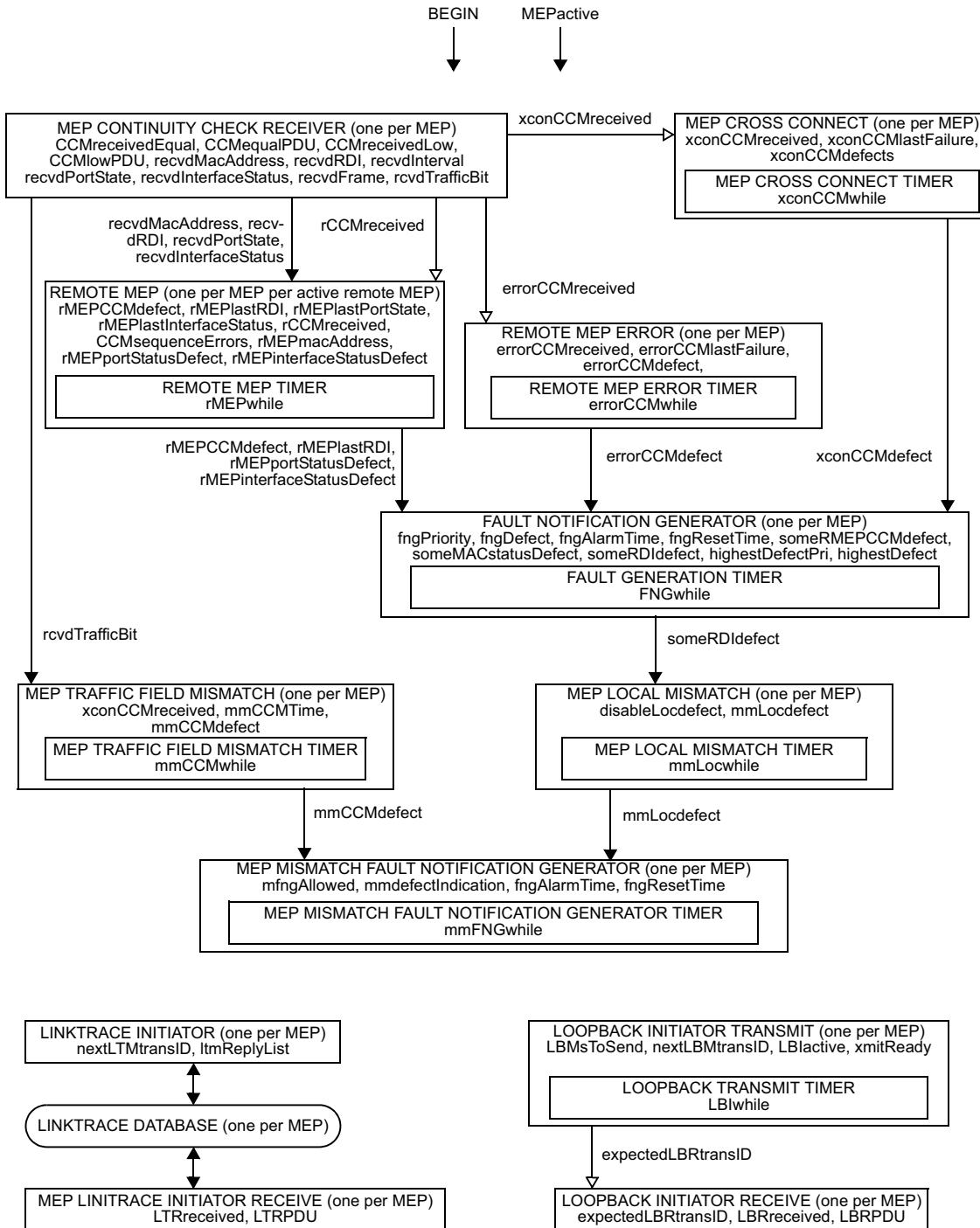


Figure 20-1—MEP state machines—overview and relationships

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 of the following shall be implemented per PBB-TE MEP implementing the Traffic field (21.6.1.4):

- g) mmCCMwhile (20.5.7)
- h) mmLocwhile (20.5.8)
- i) mmFNGwhile (20.5.9)

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):

- j) rMEPwhile (20.5.10).

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 CCMininterval 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 mmCCMwhile

Timer counter for declaring mismatch defects based on the received Traffic field. mmCCMwhile has a granularity finer than or equal to 1 ms.

20.5.8 mmLocwhile

Timer counter for declaring mismatch defects based on inconsistencies in locally declared values (20.9.9). mmLocwhile has a granularity finer than or equal to 1 ms.

20.5.9 mmFNGwhile

A timer variable used by the MEP Mismatch Fault Notification Generator state machine (20.40) to wait for defects to stabilize and disappear.

20.5.10 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 $(3.25 * \text{CCMtime}(\text{CCMinterval}))$ seconds of the receipt of a CCM, and shall set rMEPCCMdefect within $(3.5 * \text{CCMtime}(\text{CCMinterval}))$ 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 $(3.5 * \text{CCMtime}(\text{CCMinterval}))$ seconds, and a minimum of $(3.25 * \text{CCMtime}(\text{CCMinterval}))$ 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, 20.5.10).

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 nonzero 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).

The following variables may be present only in PBB-TE MAs, are local to a single PBB-TE MEP and are accessible by more than one state machine:

- h) presentTraffic (20.9.8)
- i) presentmmLoc (20.9.9)

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 either if they are enabled, or if the MEP is not associated with a single value of the Port State and VID 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 VID's member set (8.8.10), 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 either that 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. In the case of a MEP associated with a PBB-TE MA, this value is not directly configurable but it takes the value of the ESP-VID field in the 3-tuple of the PBB-TE MA's ESP that has the MEP's MAC address in its ESP-SA field.

20.9.8 presentTraffic

A Boolean value indicating if at least one Backbone Service instance is configured to use the TESI's ESP upon which this PBB-TE MEP is transmitting CCMs. presentTraffic is TRUE if and only if the backbone service instance table, of the CBP associated with this MEP, contains an entry that has in its B-VID and Default Backbone Destination fields the values of ESP-VID and ESP-DA of the monitored TESI's ESP which originates at the MEP.

20.9.9 presentmmLoc

presentmmLoc is the logical AND of presentRDI (20.9.6) and presentTraffic (20.9.8).

20.10 MEP Continuity Check Initiator variables

The following variables are local to the MEP Continuity Check Initiator state machine:

- a) CClenabled (20.10.1);
- b) CCIsentCCMs (20.10.2); and
- c) MACstatusChanged (20.10.3).

20.10.1 CClenabled

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 8-13 corresponding to the MEP's MD Level. In the case of a PBB-TE associated MEP, the destination_address parameter is set to the MAC address indicated by the ESP-DA field of the ESP having in its ESP-SA field the MAC address of the MEP;
- 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.
- o) Only in the case of a PBB-TE MEP, optionally fills the Traffic field (21.6.1.4) with the presentTraffic variable (20.9.8)

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.

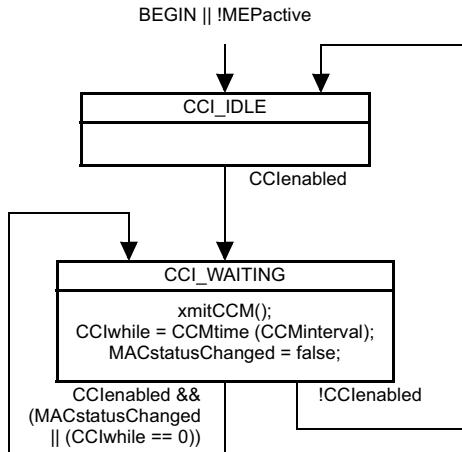


Figure 20-2—MEP Continuity Check Initiator state machine

20.13 MHF Continuity Check Receiver variables

The following variables are defined for the MHF Continuity Check Receiver:

- a) MHFrecvCCM (20.13.1); and
- b) MHFCCMPDU (20.13.2).

20.13.1 MHFrecvCCM

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:

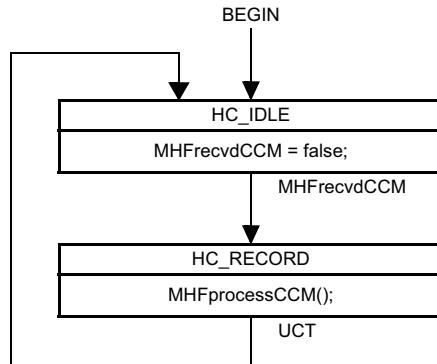
- a) MHFprocessCCM() (20.14.1).

20.14.1 MHFprocessCCM()

MHFprocessCCM() shall validate the received CCM according to 20.51.4 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) recvMacAddress (20.16.5);
- f) recvRDI (20.16.6);
- g) recvInterval (20.16.7);
- h) recvPortState (20.16.8);
- i) recvInterfaceStatus (20.16.9);
- j) recvSenderId (20.16.10);
- k) recvFrame (20.16.11); and
- l) CCMsequenceErrors (20.16.12).

The following variable is present only in PBB-TE MEP that supports the Traffic field (21.6.1.4), is local to a single PBB-TE MEP and is accessible by the MEP Mismatch state machines (20.26):

- m) rcvdTrafficBit (20.16.13)

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 recvMacAddress

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 recvRDI

Boolean flag. Set by MEPprocessEqualCCM() according to the RDI field of the last-received valid CCM.

20.16.7 recvInterval

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 recvPortState

(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. Set by MEPprocessEqualCCM().

20.16.9 recvInterfaceStatus

(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. Set by MEPprocessEqualCCM().

20.16.10 recvSenderId

(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. Set by MEPprocessEqualCCM().

20.16.11 recvFrame

Octet string holding a frame received by MEPprocessEqualCCM(). recvFrame 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 recvFrame indicates the size of the reconstructed frame. When set to the value NULL, the size of recvFrame 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].

20.16.13 recvTrafficBit

Boolean flag which is applicable only for PBB-TE MEPs supporting the Traffic field (21.6.1.4). Set by MEPprocessEqualCCM() according to the Traffic field of the last-received valid CCM.

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 20.51.4.2, and may validate the CCM according to 20.51.4.3, 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 MEP [item a) in 12.14.1.2.2, item b) in 12.14.5.3.2], then MEPprocessEqualCCM() sets xconCCMReceived (20.23.1) true (this procedural step being optional in the case of a PBB-TE MEP), reconstructs the frame containing the CCM into recvFrame, and places a timer counter value into recvInterval 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 recvFrame, and places a timer counter value into recvInterval corresponding to the value of the CCM Interval field in the received CCM.
- d) Otherwise, MEPprocessEqualCCM():
 - 1) Copies the source_address parameter into recvMacAddress;
 - 2) Copies the RDI field to recvRDI;
 - 3) Optionally copies the Port Status TLV to recvPortState;
 - 4) Optionally copies the Interface Status TLV to recvInterfaceStatus;
 - 5) Optionally copies the Sender ID TLV to recvSenderId;
 - 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.
 - 8) Only in the case of a PBB-TE MEP, optionally copies the Traffic field (21.6.1.4) to recvTrafficBit (20.16.13).

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 20.51.4.2, and may validate the CCM according to 20.51.4.3, and discard any frames that fail the validation.
- b) Otherwise, MEPprocessLowCCM() sets xconCCMreceived (20.23.1) true, reconstructs the frame containing the CCM into recvFrame, and places a timer counter value into recvInterval 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.

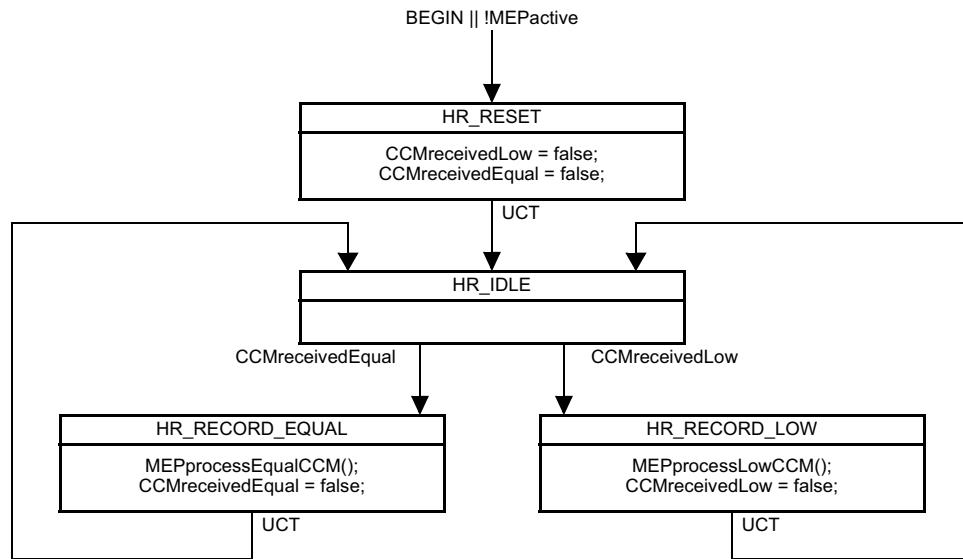


Figure 20-4—MEP Continuity Check Receiver state machine

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 $(3.25 * \text{CCMtime}(\text{CCMinterval}))$ 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 that 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).

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.

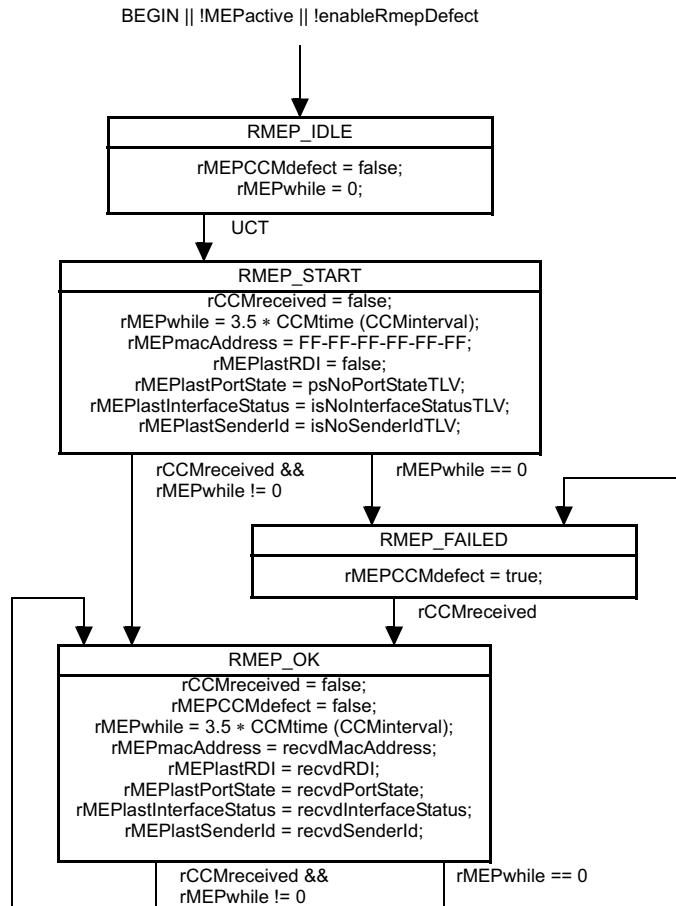


Figure 20-5—Remote MEP state machine

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 recvFrame. 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.

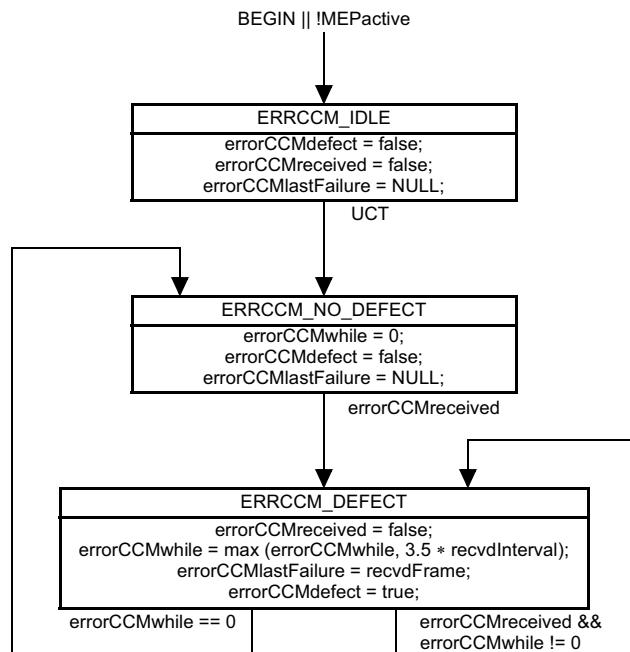


Figure 20-6—Remote MEP Error state machine

20.23 MEP Cross Connect variables

The following variables are local to the MEP Cross Connect state machine:

- xconCCMreceived (20.23.1);
- xconCCMlastFailure (20.23.2); and
- 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 recvFrame. 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.

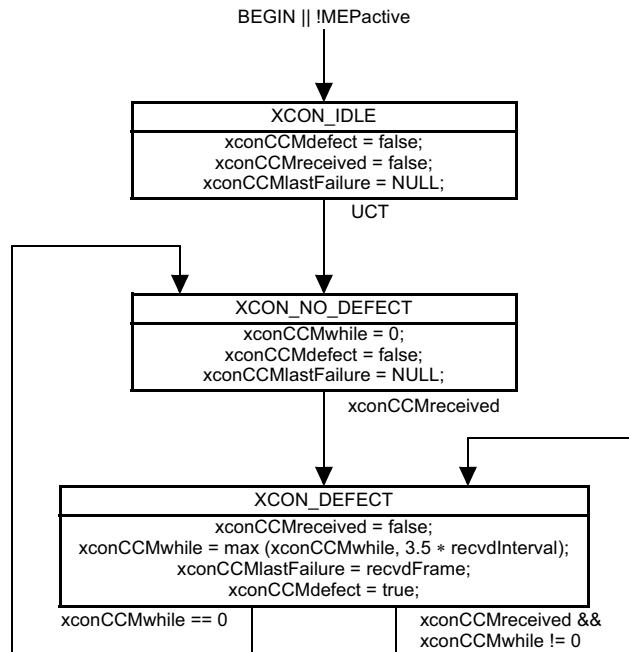


Figure 20-7—MEP Cross Connect state machine

20.25 MEP Mismatch variables

The following variables are local to the MEP Mismatch state machines for a PBB-TE MEP implementing the Traffic field (21.6.1.4):

- a) mmCCMreceived (20.25.1)
- b) mmCCMdefect (20.25.2)
- c) mmCCMTIME (20.25.3)
- d) disableLocdefect (20.25.4)
- e) mmLocdefect (20.25.5)

20.25.1 mmCCMreceived

Boolean flag set to TRUE when rcvdTrafficBit (20.16.13) does not match the presentTraffic (20.9.8).

20.25.2 mmCCMdefect

A Boolean flag set and cleared by the MEP Mismatch state machines to indicate that one or more CCMs with Traffic fields not matching the presentTraffic (20.9.8) have been received, over a period that is greater than 3.5 times the configured CCM transmission rate and given by the mmCCMTime (20.25.3). This variable is readable as a managed object [item ah] in 12.14.7.1.3].

20.25.3 mmCCMTime

The time used to initiate the mmCCMwhile timer. This is the time taken for a remote IB-BEB to send a CCM with the correct information in the Traffic field and is tied to the target protection switching time, i.e., 50 ms, as a mismatch would be normal prior to the completion of a protection switch. In the most generic case it is equal to the Detection Time + Restoration Procedure Time + Restoration Transfer time. This gives a time of $3.5 \times \text{CCM interval} + \text{Restoration Procedure Time} + 1.25 \times \text{CCM interval}$. As a result its value is set to max (50 ms, $4.75 \times \text{CCMinterval}$ + 10 ms).

20.25.4 disableLocdefect

A Boolean that disables the indication of the mmLocdefect. The variable is always set to TRUE for PBB-TE MAs that are not part of a TE protection group. On a PBB-TE MEP associated with the working entity in one or more TE protection groups to which its monitoring TESI MA is assigned, disableLocdefect is TRUE if and only if either p.SF (26.10.3.3.2) or LoP (26.10.3.3.4) is TRUE on any of these TE protection groups. On a PBB-TE MEP associated with only one protection entity in all the TE protection groups to which it is assigned, disableLocdefect is TRUE if and only if FS (26.10.3.3.5) is TRUE on this TE protection group.

20.25.5 mmLocdefect

A Boolean flag set and cleared by the MEP Local Mismatch state machine to indicate that presentmmLoc (20.9.9) is set to TRUE, over a period of 50 ms [item ai] in 12.14.7.1.3].

20.26 MEP Mismatch state machines

The MEP Mismatch state machines implement the functions specified by the state diagrams in Figure 20-8, Figure 20-9, and the variable declarations in 20.23. There is one MEP Traffic Field Mismatch state machine and one MEP Local Mismatch state machine per PBB-TE MEP implementing the Traffic field (21.6.1.4).

20.27 MP Loopback Responder variables

The following variables are local to the MP Loopback Responder state machine:

- a) LBMreceived (20.27.1); and
- b) LBMPDU (20.27.2).

20.27.1 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.

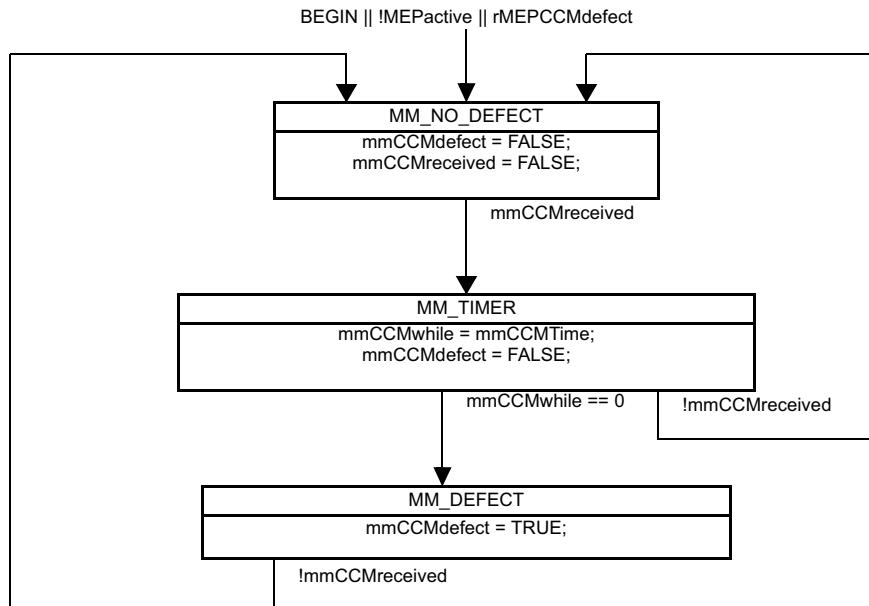


Figure 20-8—MEP Traffic Field Mismatch state machine

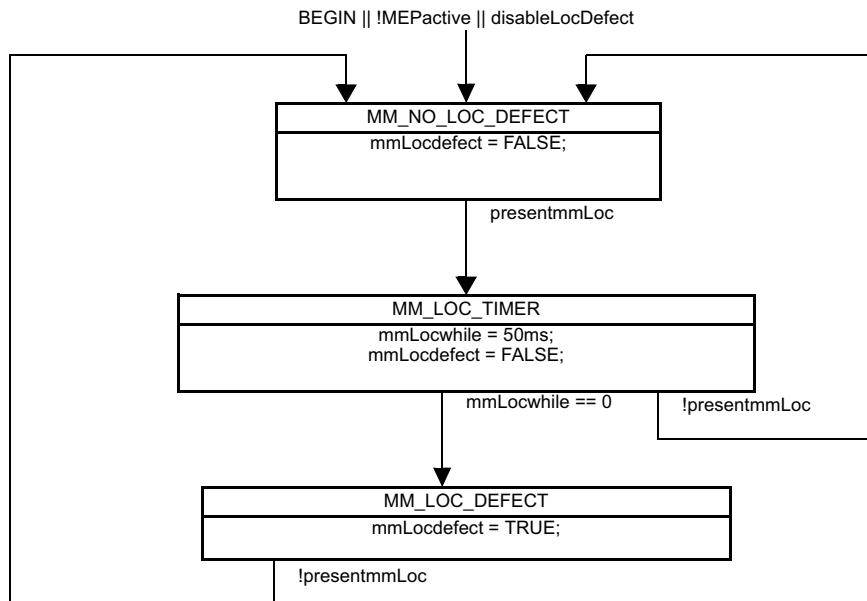


Figure 20-9—MEP Local Mismatch state machine

20.27.2 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.

20.28 MP Loopback Responder procedures

The following procedures are defined for the Loopback Responder:

- a) ProcessLBM() (20.28.1); and
- b) xmitLBR() (20.28.2).

20.28.1 ProcessLBM()

Called by the MP Loopback Responder state machine whenever an LBM is received. ProcessLBM() processes the LBM in LBMPDU as follows:

- a) If
 - 1) the destination_address parameter contains an Individual MAC address that does not match the MAC address of the receiving MP; and
 - 2) the MP Loopback Responder state machine does not reside in a PBB-TE MHF;
 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 a non-PBB-TE associated 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 20.51.4.2, and may validate the LBM according to 20.51.4.3, and discard any frames that fail the validation;
- e) If the MP Loopback Responder state machine resides in a PBB-TE MEP and the LBM carries a PBB-TE MIP TLV, ProcessLBM() discards the LBM and performs no further processing;
- f) If the MP Loopback Responder state machine resides in a PBB-TE MHF and;
 - 1) There is no PBB-TE MIP TLV; or
 - 2) There is a PBB-TE MIP TLV but the address carried in the MIP MAC address field does not match the MAC address of the receiving MIP;
 ProcessLBM() forwards the LBM to the Passive MHF Multiplexer and performs no further processing;
- g) Otherwise, ProcessLBM() calls xmitLBR() to generate and transmit an LBR.

20.28.2 xmitLBR()

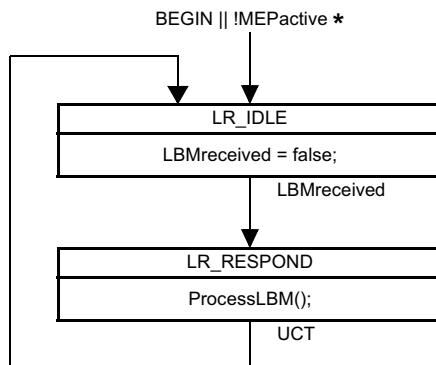
Called by ProcessLBM() to transmit an LBR. xmitLBR() constructs an LBR from the LBM contained in LBMPDU, and transmits it to the Active SAP using an M_UNITDATA.request as follows. xmitLBR():

- a) Sets the destination_address parameter to the source_address of the received LBM if the MP is not associated with a PBB-TE MA; otherwise to
 - 1) The value of the ESP-DA of the MA's ESP that has the replying MEP's MAC address [item i) in 12.14.7.1.3] in its ESP-SA field, if the MEP is a PBB-TE MEP; or otherwise to
 - 2) The value carried in the Reverse MAC field contained in the PBB-TE MIP TLV of the received LBM, if this is a Group MAC address; or otherwise to
 - 3) The source_address of the received LBM;
- b) Sets the source_address parameter to the MAC address of the replying MP if the MP is not a PBB-TE MHF; or otherwise to
 - 1) The destination_address of the received LBM if this is an Individual MAC address; or otherwise to
 - 2) The value carried in the Reverse MAC field contained in the PBB-TE MIP TLV of the received LBM;

- c) (Only for PBB-TE MAs) The `vlan_identifier` for the LBR is the Primary VID [item d) in 12.14.7.1.3] associated with the replying PBB-TE MEP; otherwise if the received LBM contains a PBB-TE MIP TLV, the value in the Reverse VID field is used as the `vlan_identifier` parameter;
- d) Changes the OpCode field (21.4.3) from LBM to LBR;
- e) Copies the remainder of the LBM's `mac_service_data_unit` verbatim to the LBR; and
- f) If the replying MP is a MEP, increments the LBR transmission counter by 1 [item ad) in 12.14.7.1.3].

20.29 MP Loopback Responder state machine

The MP Loopback Responder state machine implements the functions specified by the state diagram in Figure 20-10, using the variables in 20.27 and the procedures in 20.28. Although the definition of the MP Loopback Responders in 19.2.10 and 19.3.8 requires 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.



* MEPactive in a MEP; not present (always true) in an MHF

Figure 20-10—MP Loopback Responder state machine

20.30 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` (20.30.1);
- b) `nextLBMtransID` (20.30.2);
- c) `expectedLBRtransID` (20.30.3);
- d) `LBIactive` (20.30.4);
- e) `xmitReady` (20.30.5);
- f) `LBRReceived` (20.30.6); and
- g) `LBRPDU` (20.30.7).

20.30.1 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 nonzero value starts transmission of LBMs. `LBMsToSend` is decremented by 1 by the MEP Loopback Initiator transmit state machine with each transmission.

20.30.2 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].

20.30.3 expectedLBRtransID

The value expected to be found in the Loopback Transaction Identifier field of the next LBR received by ProcessLBR(). Altered by ProcessLBR() (see 20.33.1).

20.30.4 LBactive

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 nonzero 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.

20.30.5 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.

20.30.6 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.

20.30.7 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.

20.31 MEP Loopback Initiator transmit procedures

The following procedure is local to the MEP Loopback Initiator transmit state machine:

- a) xmitLBM() (20.31.1).

20.31.1 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]. If the MEP is configured on a PBB-TE MA, the destination_address parameter is set to the MAC address indicated by the value of the ESP-DA field of the MA's ESP that has the MEP's MAC address indicated in its ESP-SA field;
- 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 (20.30.2) to the Loopback Transaction Identifier field (21.7.3) of the LBM;
- e) If the MEP is configured on a PBB-TE MA, constructs a PBB-TE MIP TLV using in the MIP MAC address field the destination address parameter from the appropriate managed object [item b) in 12.14.7.3.2] and in the Reverse VID field the parameter [item f) in 12.14.7.3.2]; if the MEP is a root in a point-to-multipoint TESI, the Reverse MAC field is also included in the PBB-TE MIP TLV carrying the ESP-SA value of any of the MA's point-to-point ESPs; if the MEP is a leaf in a point-to-multipoint TESI, the Reverse MAC field carries the ESP-DA of the point-to-multipoint ESP; the PBB-TE MIP TLV is not used if the LBM destination address in 12.14.7.3.2 is associated with any of the values in the ESP-DA field of the monitored MA's ESPs;
- f) 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 nonzero length;
- g) 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
- h) Increments nextLBMtransID (20.30.2) by 1, wrapping around from $2^{32} - 1$ to 0.

20.32 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 20-11 and the procedures in 20.31. It shares with the MEP Loopback Initiator receive state machine the variable declarations in 20.30.

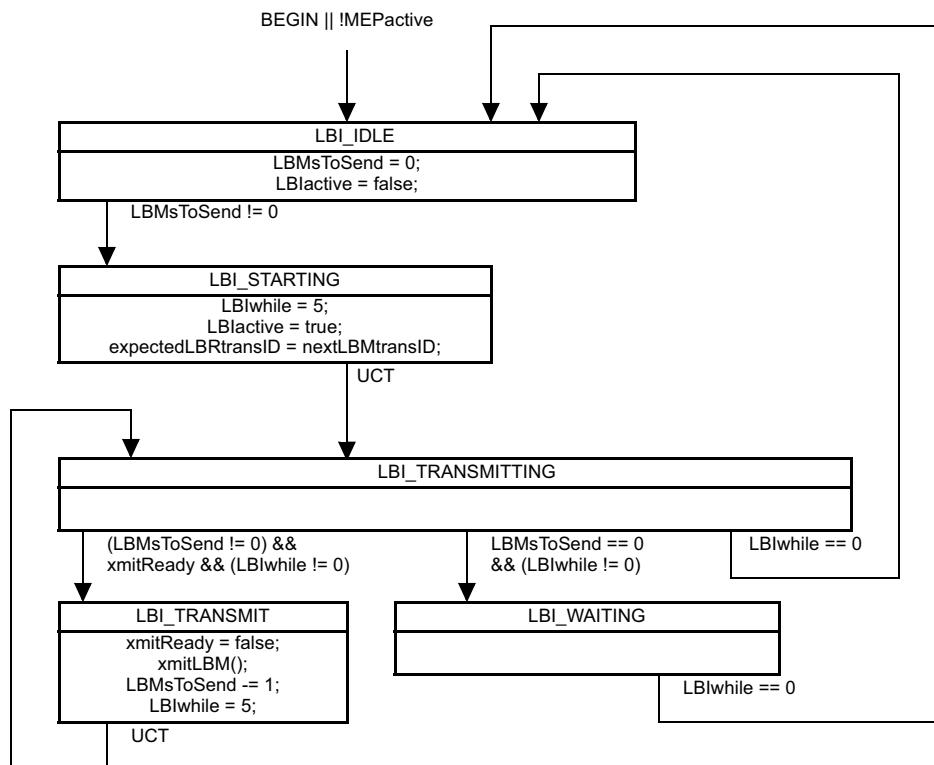


Figure 20-11—MEP Loopback Initiator transmit state machine

20.33 MEP Loopback Initiator receive procedures

The following procedure is local to the MEP Loopback Initiator receive state machine:

- a) ProcessLBR() (20.33.1).

20.33.1 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 20.51.4.2, and may validate the received LBR according to 20.51.4.3, 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 the number of correct LBRs received [item y) in 12.14.7.1.3] is incremented by 1;
 - 2) If the two values are unequal, then 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;
 - 3) Whether or not the two values were equal, expectedLBRtransID is then incremented by 1.
- d) ProcessLBR() may perform a bit-by-bit 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.

20.34 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 20-12 and the procedures in 20.33. It shares with the MEP Loopback Initiator transmit state machine the variable declarations in 20.30.

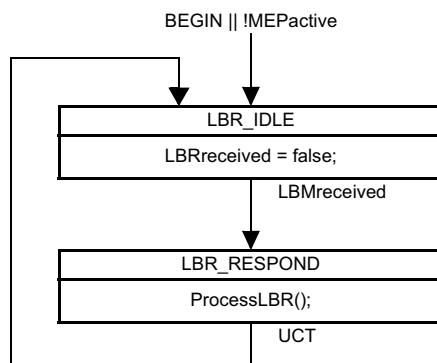


Figure 20-12—MEP Loopback Initiator receive state machine

20.35 MEP Fault Notification Generator variables

The following variables are local to the MEP Fault Notification Generator state machine:

- a) fngPriority (20.35.1);
- b) fngDefect (20.35.2);
- c) fngAlarmTime (20.35.3);
- d) fngResetTime (20.35.4);
- e) someRMEPCCMdefect (20.35.5);
- f) someMACstatusDefect (20.35.6);
- g) someRDIdefect (20.35.7);
- h) highestDefectPri (20.35.8); and
- i) highestDefect (20.35.9).

20.35.1 fngPriority

An integer specifying the priority of the last defect reported in a Fault Alarm. fngPriority takes the same values as highestDefectPri (20.35.8).

20.35.2 fngDefect

An enumerated value specifying the last defect reported in a Fault Alarm. fngDefect takes the same values as highestDefect (20.35.9).

20.35.3 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].

20.35.4 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].

20.35.5 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].

20.35.6 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 either if 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].

20.35.7 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 *rMEP_{last}RDI* 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].

20.35.8 highestDefectPri

An integer value indicating the priority of the defect named in the variable *highestDefect*. See also Table 20-1.

20.35.9 highestDefect

An enumerated value indicating the highest priority defect among the variables *xconCCMdefect* (20.23.3), *errorCCMdefect* (20.21.3), *someRMEPCCMdefect* (20.35.5), *someMACstatusDefect* (20.35.6), and *someRDIdefect* (20.35.7), 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.

20.36 MEP Fault Notification Generator procedures

The following procedure is local to the MEP Fault Notification Generator state machine:

- a) *xmitFaultAlarm()* (20.36.1).

20.36.1 xmitFaultAlarm()

Transmits a Fault Alarm (12.14.7.7). The identity of the MEP and the variable *fngDefect* (20.35.2), 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.

20.37 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 20-13, the variables in 20.35, and the procedure in 20.36. 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].

20.38 MEP Mismatch Fault Notification Generator variables

The following variables are local to the MEP Mismatch Fault Notification Generator state machine for a PBB-TE MEP implementing the Traffic field (21.6.1.4):

- a) *mfngAllowed* (20.38.1)
- b) *mmdefectIndication* (20.38.2)
- c) *mfngAlarmTime* (20.38.3)
- d) *mfngResetTime* (20.38.4)

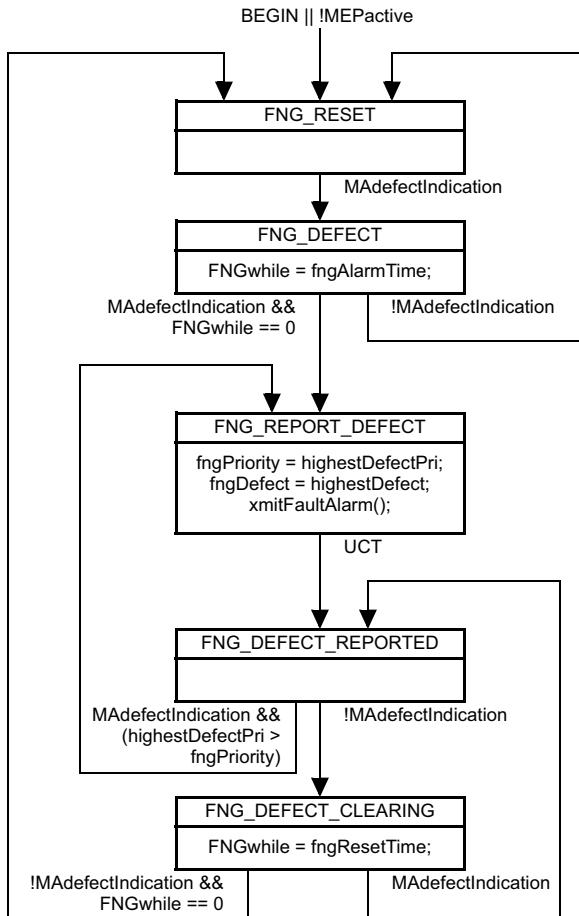


Figure 20-13—MEP Fault Notification Generator state machine

20.38.1 mfngAllowed

A Boolean indicating the mismatch defect is allowed to generate a Fault Alarm. Its value is configurable as a managed object [item ag] in 12.14.7.1.3].

20.38.2 mmdefectIndication

A Boolean indicating the presence of the mismatch defect that is allowed to generate a Fault Alarm. It corresponds to the logical AND of the variables mfngAllowed (20.38.1), mmCCMdefect (20.25.2), and mmLocdefect (20.25.5).

20.38.3 mfngAlarmTime

The time that one or more defects must be present before a mismatch Fault Alarm is issued. Default value 2.5 s. Also a managed object [item l] in 12.14.7.1.3].

20.38.4 mfngResetTime

The time, after a mismatch Fault Alarm, that no defects must be present before another mismatch Fault Alarm is enabled. Default value 10 s. Also a managed object [item m] in 12.14.7.1.3].

20.39 MEP Mismatch Fault Notification Generator procedures

The following procedure is local to the MEP Mismatch Fault Notification Generator state machine for a PBB-TE MEP implementing the Traffic field (21.6.1.4):

- a) xmitFaultAlarm() (20.36.1)

20.39.1 xmitFaultAlarm()

Transmits a Fault Alarm (12.14.7.7). The identity of the MEP and a value indicating that the cause of the Fault Alarm is a mismatch defect, are transmitted in the Fault Alarm PDU. The format and method of transmission of the Fault Alarm is not specified in this standard.

20.40 MEP Mismatch Fault Notification Generator state machine

A PBB-TE MEP implementing the Traffic field (21.6.1.4) creates a single instance of the MEP Mismatch Fault Notification Generator state machine. The MEP Mismatch Fault Notification Generator state machine implements the function specified by the state diagram in Figure 20-14, the variables in 20.38, and the procedure in 20.39. The current state of the MEP Mismatch Fault Notification Generator state machine is available in a managed object [item ak] in 12.14.7.1.3].

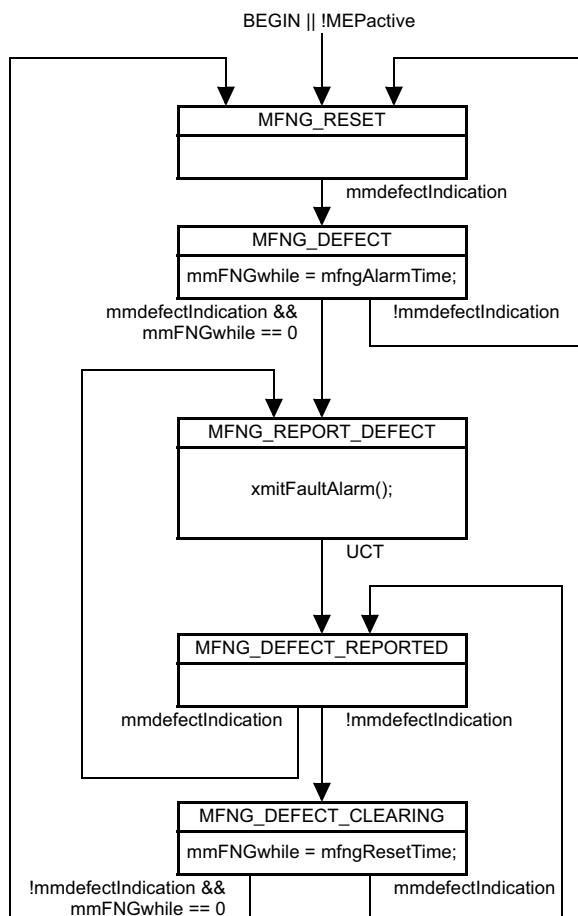


Figure 20-14—MEP Mismatch Fault Notification Generator state machine

20.41 MEP Linktrace Initiator variables

The following variables are local to the MEP Linktrace Initiator variables:

- a) nextLTMtransID (20.41.1); and
- b) ltmReplyList (20.41.2).

20.41.1 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].

20.41.2 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 (20.41.2.1);
- b) ltrReplyTTL (20.41.2.2);
- c) ltrLastEgressId (20.41.2.3);
- d) ltrNextEgressId (20.41.2.4);
- e) ltrRelayAction (20.41.2.5);
- f) ltrIngressAction (20.41.2.6);
- g) ltrIngressAddress (20.41.2.7);
- h) ltrIngressPortIdSubtype (20.41.2.8);
- i) ltrIngressPortId (20.41.2.9);
- j) ltrEgressAction (20.41.2.10);
- k) ltrEgressAddress (20.41.2.11);
- l) ltrEgressPortIdSubtype (20.41.2.12);
- m) ltrEgressPortId (20.41.2.13);
- n) ltrSenderIdTlv (20.41.2.14); and
- o) ltrOrgSpecTlv (20.41.2.15).

20.41.2.1 ltrFlags

The bit string returned in the Flags field (21.9.1) of the LTR, including the FwdYes and TerminalMEP bits.

20.41.2.2 ltrReplyTTL

The integer value returned in the Reply TTL field (21.9.4) of the LTR.

20.41.2.3 ltrLastEgressId

The octet string returned in the Last Egress Identifier field (21.9.7.1) of the LTR Egress Identifier TLV.

20.41.2.4 ltrNextEgressId

The integer value returned in the Next Egress Identifier field (21.9.7.2) of the LTR Egress Identifier TLV.

20.41.2.5 ltrRelayAction

The enumerated value returned in the Relay Action field (21.9.5) of the LTR. The enumerated values are listed in Table 21-28.

20.41.2.6 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 21-31. An enumerated value of 0 indicates that no Reply Ingress TLV was returned in the LTR.

20.41.2.7 ltrIngressAddress

The MAC address of the MP on the Ingress Port. Contents are undefined if no Reply Ingress TLV was returned in the LTR.

20.41.2.8 ltrIngressPortIdSubtype

An enumerated value as specified by IEEE Std 802.1ABTM, indicating the format of ltrIngressPortId. Contents are undefined if no Reply Ingress TLV was returned in the LTR.

20.41.2.9 ltrIngressPortId

An octet string as specified by IEEE Std 802.1AB, identifying the Ingress Port, in the format identified in ltrIngressPortIdSubtype.

20.41.2.10 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 21-33. A value of 0 indicates that no Reply Egress TLV was returned in the LTR.

20.41.2.11 ltrEgressAddress

The MAC address of the MP on the Egress Port. Contents are undefined if no Reply Egress TLV was returned in the LTR.

20.41.2.12 ltrEgressPortIdSubtype

An enumerated value as specified by IEEE Std 802.1AB, indicating the format of ltrEgressPortId. Contents are undefined if no Reply Ingress TLV was returned in the LTR.

20.41.2.13 ltrEgressPortId

An octet string as specified by IEEE Std 802.1AB, identifying the Egress Port, in the format identified in ltrEgressPortIdSubtype.

20.41.2.14 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.

20.41.2.15 ltrOrgSpecTlv

An octet string containing the Organization-Specific TLVs (21.5.2), if any were present in the LTR.

20.42 MEP Linktrace Initiator procedures

The following procedure is local to the MEP Linktrace Initiator:

- a) xmitLTM() (20.42.1).

20.42.1 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 8-14 corresponding to the MEP's MD Level. If the MEP is configured on a PBB-TE MA, the destination_address parameter is the MAC address indicated by the value of the ESP-DA field of the MA's ESP that has the MEP's MAC address indicated in its ESP-SA field;
- 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) If the MEP is configured on a PBB-TE MA, constructs a PBB-TE MIP TLV using in the MIP MAC address field a null field and in the Reverse VID field the parameter [item e] in 12.14.7.4.2]; if the MEP is a root in a point-to-multipoint TESI, the Reverse MAC field is also included in the PBB-TE MIP TLV carrying the ESP-SA value of any of the MA's point-to-point ESPs; if the MEP is a leaf in a point-to-multipoint TESI, the Reverse MAC field carries the ESP-DA of the point-to-multipoint ESP;
- d) Sets the priority parameter to the same value as for CCMs [item h] in 12.14.7.1.3];
- e) Copies nextLTMtransID [item ab] in 12.14.7.1.3] to the LTM Transaction Identifier field (21.8.3) of the LTM;
- f) Sets the LTM Egress Identifier TLV (21.8.8) to a value that is unique among all Bridges in the Management Domain.
- g) Sets the Target MAC Address field (21.8.6) from the appropriate managed object [item c] in 12.14.7.4.2];
- h) Sets the LTM TTL field (21.8.4) from the appropriate managed object [item d] in 12.14.7.4.2];
- i) 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;
- j) 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;
- k) Creates a new entry in the ltmReplyList variable (20.41.2) for this LTM, identified by the LTM Transaction Identifier in nextLTMtransID (20.41.1); and
- l) Increments nextLTMtransID (20.41.1) 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.

20.43 MEP Linktrace Initiator receive variables

The following variables are local to the MEP Linktrace Initiator receive state machine:

- a) LTRReceived (20.43.1); and
- b) LTRPDU (20.43.2).

20.43.1 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.

20.43.2 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.

20.44 MEP Linktrace Initiator receive procedures

The following procedure is local to the MEP Linktrace Initiator receive state machine:

- a) ProcessLTR() (20.44.1).

20.44.1 ProcessLTR()

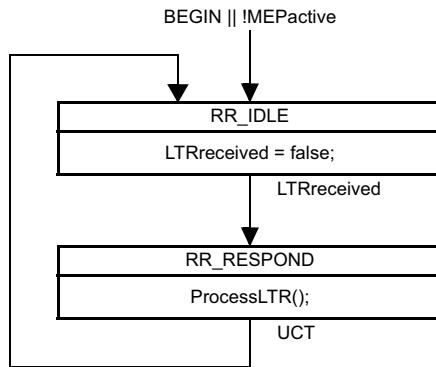
The ProcessLTR() procedure is called by the MEP Linktrace Initiator receive state machine whenever an LTR is received, and processes the LTR contained in the LTRPDU variable as follows:

- 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 20.51.4, 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.

20.45 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 20-15, the variables in 20.43, and the procedures in 20.44.

**Figure 20-15—MEP Linktrace Initiator receive state machine**

20.46 Linktrace Responder variables

The following variables are local to the Linktrace Responder:

- a) `nPendingLTRs` (20.46.1);
- b) `LTMreceived` (20.46.2); and
- c) `LTMPDU` (20.46.3).

20.46.1 nPendingLTRs

An integer value used to track the number of LTRs that have been enqueued for transmission by `enqueueLTR()` and not yet transmitted by `xmitOldestLTR()`. Can be reset to 0 by `clearPendingLTRs()`.

20.46.2 LTMreceived

A Boolean value set when a valid LTM is received and cleared by the LTM Receiver state machine.

20.46.3 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.

20.47 LTM Receiver procedures

The procedures local to the LTM Receiver state machine are:

- a) `ProcessLTM()` (20.47.1);
- b) `clearPendingLTRs()` (20.47.2);
- c) `ForwardLTM()` (20.47.3); and
- d) `enqueueLTR()` (20.47.4).

20.47.1 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 `enqueueLTR()` 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 20.51.3. If the LTM is invalid, no further validation steps are performed.
- b) 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.
- c) Otherwise, if the *vlan_identifier* identifies an ESP-VID the LTM is valid, and no further validation steps are performed.
- d) Otherwise, if the *destination_address* parameter of the LTM is a Group address, the LTM is valid, and no further validation steps are performed.
- e) 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.
- f) Otherwise, the LTM is invalid.

20.47.1.1 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 20-16 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.

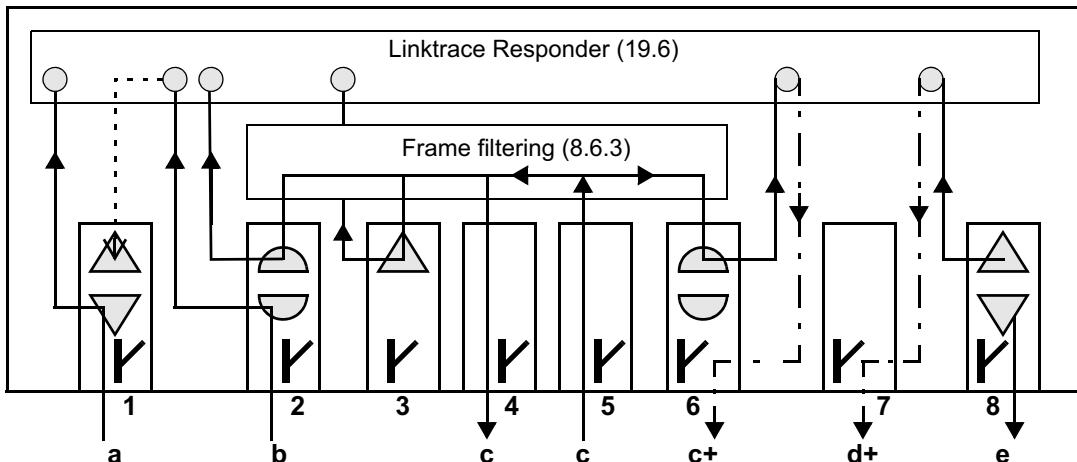


Figure 20-16—Linktrace Responder, MEPs, MHFs, and LOMs

NOTE—All of the MPs shown in Figure 20-16 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 if 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 (20.50).

20.47.1.2 Ingress Port, *vlan_identifier*, and Egress Port determination

In 20.3.2 and Figure 20-16, 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 (6.6.1) 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, or in the case of an MP associated with a PBB-TE MA, the destination_address of 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. In the case of an MP associated with a PBB-TE MA, the query uses the destination_address and the *vlan_identifier* of the LTM as the corresponding parameters for the lookup. 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. If the resultant set contains more than one Bridge Port, and the MP is associated with a PBB-TE MA, those Bridge Ports are all Egress Ports, and step b is not performed.

NOTE 1—If there are only two Bridge Ports that are members of the VID'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 by non-PBB-TE MHFs. PBB-TE MHFs associated with point-to-multipoint TESIs forward LTMs even if multiple Egress Ports are identified.

20.47.1.3 LTM is received by a Down MEP

In case a) in 20.47.1.1, illustrated in Figure 20-16, 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 enqueueLTR() (20.47.4) 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 20.47.1.2. 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 enqueueLTR() (20.47.4) to enqueue an LTR for the Linktrace SAP through which the LTM was received. The LTM is discarded, and no further processing takes place.

20.47.1.4 LTM is received by a Down MHF or originated by an Up MEP

In case b) in 20.47.1.1, illustrated in Figure 20-16, the LTM is received by a Down MHF. In case d) in 20.47.1.1, illustrated in Figure 20-16, 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 enqueueLTR() (20.47.4) 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 20.47.1.2. If an 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 enqueueLTR() (20.47.4) 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() (20.47.3) 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 enqueueLTR() (20.47.4) 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 enqueueLTR() (20.47.4) 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() (20.47.3) 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.

20.47.1.5 LTM is received by an Up MEP

In case c) in 20.47.1.1, illustrated in Figure 20-16, 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 20-16. 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 enqueueLTR() (20.47.4) 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 20.47.1.2. 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 enqueueLTR() (20.47.4) to enqueue an LTR for the MEP Linktrace SAP of the receiving Up MEP. The LTM is then discarded.

20.47.1.6 LTM is received by an Up MHF

In case c) in 20.47.1.1, illustrated in Figure 20-16, 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 20-16. 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 enqueueLTR() (20.47.4) 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 20.47.1.2. 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 enqueueLTR() (20.47.4) 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() (20.47.3) 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.

20.47.2 clearPendingLTRs()

Clears the queue of pending LTRs for this MP. Resets nPendingLTRs to 0.

20.47.3 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) In PBB-TE related MAs, uses the destination_address, source_address and vlan_identifier of the input LTM and sets the drop_eligible parameter of the forwarded LTM to FALSE;
- f) 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
- g) Transmits the forwarded LTM on the specified SAP.

20.47.4 enqueueLTR()

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, enqueueLTR():

- a) Uses the MAC address contained in the LTM's Original MAC Address field as the destination_address of the LTR. If the MP is associated with a PBB-TE MA, the LTR uses as destination_address:
 - 1) The value carried in the Reverse MAC field, contained in the PBB-TE MIP TLV of the received LTM, if this is a Group MAC address; otherwise:
 - 2) The source_address of the received LTM;
- b) Uses the MAC address of the MP owning the SAP specified for output as the source_address of the LTR, if the MP is not a PBB-TE MHF; otherwise the LTR uses as the source_address:
 - 1) The value of the destination_address of the received LTM if this is an Individual MAC address; otherwise:
 - 2) The value carried in the Reverse MAC field, contained in the PBB-TE MIP TLV of the received LTM;
- 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. In PBB-TE related MA's, sets the vlan_identifier parameter as the value carried in the Reverse VID field contained in the PBB-TE MIP TLV of the received LTM;
- 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 21-28;
- 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 21-31;
 - 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 21-33;
 - 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.

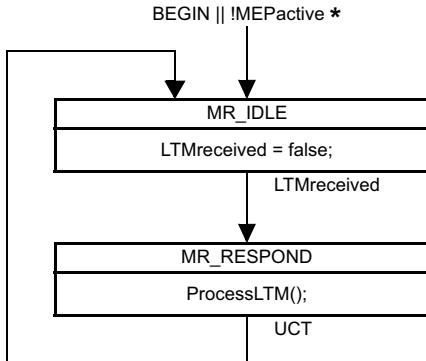
20.48 LTM Receiver state machine

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 20-17, the variables in 20.46, and the procedures in 20.47.

20.49 LTR Transmitter procedure

The procedure local to the LTR Transmitter state machine is:

- a) xmitOldestLTR() (20.49.1).



* MEPactive in a MEP; not present (always true) in an MHF

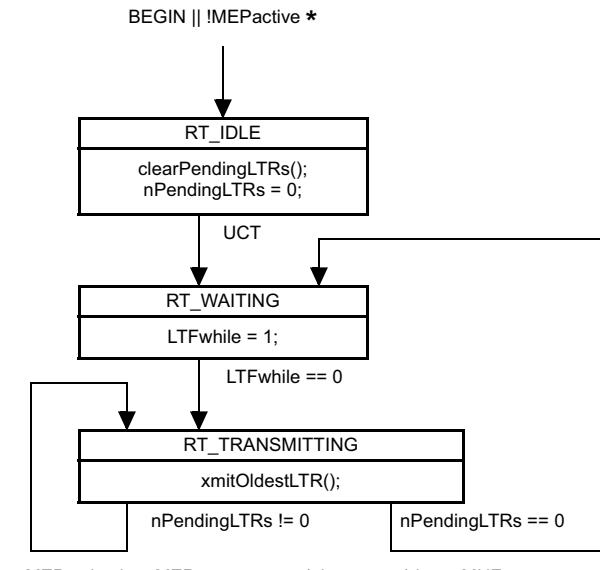
Figure 20-17—LTM Receiver state machine

20.49.1 `xmitOldestLTR()`

If and only if `nPendingLTRs` is nonzero, dequeues a single LTR and transmits it, and decrements `nPendingLTRs` by 1.

20.50 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 20-18, the variables in 20.46, and the procedures in 20.47.



* MEPactive in a MEP; not present (always true) in an MHF

Figure 20-18—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.

20.51 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.

20.51.1 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, nonstandard 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.

20.51.2 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 (02/2008), 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.

20.51.3 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.

20.51.4 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 (20.51.4.1);
- b) Operations required of all MP components when performing the Validation pass (20.51.4.2); and
- c) Operations that are required of some MP components, and optional for others (20.51.4.3).

20.51.4.1 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.

20.51.4.2 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 20.51.2.

20.51.4.3 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 20.51.4.2.
- 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**:

20.51.5 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.

20.51.6 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 20.51.2, 20.51.4, and/or 20.51.5. The Organization-Specific TLV is the only available compliant extension to this standard.

20.52 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.

20.53 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* (20.30.2, for LBM), and *nextLTMtransID* (20.41.1, 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.

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), Linktrace Reply (LTR, 21.9), Send Frame Message (SFM, 29.4.3), and the Reflected Frame Message (RFM, 29.4.2).

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 LSB 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 Length/Type field, e.g., IEEE 802.3 media, the identification consists of two octets containing the EtherType 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: Length/Type 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 8-13 as the destination_address. LTMs monitoring a VID-distinguished service instance use those shown in Table 8-14. 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. CFM PDUs belonging to MAs monitoring service instances distinguished by their I-SID use the same set of destination addresses as those belonging to MAs monitoring VLAN service instances. MAs monitoring TESIs are distinguished from MAs monitoring VLAN services by the vlan_identifier parameters and from each other by the 3-tuples <ESP-DA, ESP-SA, ESP-VID> of their ESPs. CFM PDUs belonging to PBB-TE MAs use the MAC address identified in the ESP-DA fields of their ESP identifiers.

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

Field	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.

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).

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
Reflected Frame Message (RFM)	6
Send Frame Message (SFM)	7
Reserved for IEEE 802.1	8 – 31
Defined by ITU-T Y.1731 (02/2008)	32 – 63
Reserved for IEEE 802.1	64 – 255

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

Field	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.

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
PBB-TE MIP TLV	9
Data Part 1 TLV (29.3.3.2)	10
Data Part 2 TLV (29.3.3.2)	11
Truncated Data TLV (29.3.3.2)	12
Reserved for IEEE 802.1	13 – 30
Organization-Specific TLV	31
Defined by ITU-T Y.1731 (02/2008)	32 – 63
Reserved for IEEE 802.1	64 – 255

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.³² 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.

Table 21-7—Organization-Specific TLV format

Field	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:

³² Interested applicants should contact the IEEE Standards Department, Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854-4141, USA, <http://standards.ieee.org/regauth/index.html>.

- 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 20.51.

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 IEEE Std 802.1AB (Station and Media Access Control Connectivity Discovery), 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].

Table 21-8—Sender ID TLV format

Field	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)

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.

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 either is 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. 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 (1999)). 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 value 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) [B41].

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.

Validation Test: The Port Status TLV contains one of the values listed in Table 21-10.

Table 21-9—Port Status TLV format

Field	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

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

Field	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 reflected data frame in RFM, to be decapsulated data in SFM, the transmission of different frame sizes to test Maximum Service Data Unit Size capabilities and the testing for data-specific error dependencies. The Data TLV may be included in the LBM and the LBR, the RFM, and the SFM, 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

Field	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

Field	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:

- 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.

Table 21-15—Continuity Check Message format

Field	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 (02/2008) (02/2008)	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.

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 MSB 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, the Traffic field, and the CCM Interval field are set to 0 by the transmitting MP, and are not to be examined by the receiving MP [item b) in 20.51.2].

21.6.1.3 CCM Interval

The least significant 3 bits of the Flags field constitute the CCM Interval field. The CCM Interval field is encoded as specified in Table 21-16.

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.

21.6.1.4 Traffic field

In the case of a PBB-TE MA, the second most significant bit of the Flags field is the Traffic bit. This bit if supported, is used to indicate the presence of backbone service instances in the monitored TESI. This bit is set to 1 if the transmitting MEP's presentTraffic variable (20.9.8) is set, and 0 if not. This field is not examined if the receiving MPs are not PBB-TE MEPs supporting the traffic field.

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

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 either transmits 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

Field	Octet
Maintenance Domain Name Format (not 1)	1
Maintenance Domain Name Length	2
Maintenance Domain Name	$3 - (\text{mdnl}^{\text{a}} + 2)$
Short MA Name Format	$(\text{mdnl} + 3)$
Short MA Name Length	$(\text{mdnl} + 4)$
Short MA Name	$(\text{mdnl} + 5) - (\text{mdnl} + \text{smanl}^{\text{b}} + 4)$
0 pad, if necessary	$(\text{mdnl} + \text{smanl} + 5) - 48$

^a Maintenance Domain Name Length.^b Short MA Name Length.**Table 21-18—CCM MAID field format: Maintenance Domain not present**

Field	Octet
Maintenance Domain Name Format (1)	1
Short MA Name Format	2
Short MA Name Length	3
Short MA Name	$4 - (\text{smanl}^{\text{a}} + 3)$
0 pad, if necessary	$(\text{smanl} + 4) - 48$

^a 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 21-19—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 (02/2008)	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
ICC-based format as specified in ITU-T Y.1731 (02/2008)	32
Defined by ITU-T Y.1731	33 – 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 (02/2008)

(16 octets) The use of this field is defined by ITU-T Y.1731. 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

Field	Octet
Common CFM Header	1 – 4
Loopback Transaction Identifier	5 – 8
Reserved for definition in future versions of the protocol ^a	
Additional LBM/LBR TLVs	First TLV Offset + 5 ^b
End TLV (0)	First TLV Offset + 5, if no Additional 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 (20.30.2) to this field.

21.7.4 Additional LBM/LBR TLVs

The following TLV shall be included in every LBM transmitted by a MEP to a MIP configured on a PBB-TE MA as specified in 21.7.5. The TLV will not be present in any other type of MA.

- a) PBB-TE MIP TLV (21.7.5).

The following TLV should be included in every LBM transmitted:

- b) Sender ID TLV (21.5.3).

The following TLV can be included by management action in any LBM transmitted:

- c) Data TLV (21.5.6).

The following TLV may be included in any LBM transmitted:

- d) Organization-Specific TLV (21.5.2).

The PBB-TE MIP TLV, if present, shall be placed first. The other Additional LBM/LBR TLVs may be placed in any order by the MEP transmitting the LBM.

21.7.5 PBB-TE MIP TLV

Type = 9. Used only in MAs monitoring PBB-TE services. It enables intermediate MIPs to selectively intercept LBMs that are targeting them. The PBB-TE MIP TLV is not used if the target address of the LBM or LTM is associated with any of the values in the ESP-DA field of the monitored MA's ESPs but otherwise it shall be present; The format of the PBB-TE MIP TLV is illustrated in Table 21-22.

Table 21-22—PBB-TE MIP TLV format

Field	Octet
Type = 9	1
Length	2–3
MIP MAC address	4–9
Reverse VID	10–11
Reverse MAC	12–17

Validation Test: The MIP MAC address field contains an Individual MAC address.

Validation Test: The Reverse MAC field contains an Individual, or a Group MAC address as specified in 21.7.5.3.

Validation Test: The Length field either is 8, indicating that no Reverse MAC field is present, or is 14, and thus contains the Reverse MAC field.

21.7.5.1 MIP MAC address

The MIP MAC address contains the destination MAC address for LBMs transmitted by the MEP [item b) in 12.14.7.3.2]. In the case of LTMs, this field is null.

21.7.5.2 Reverse VID

The Reverse VID field is used by the MIPs in order to set the `vlan_identifier` parameters on LBRs and LTRs. Its value is provided by the operator on the initiation of the LBM [item f) in 12.14.7.3.2] or the LTM [item e) in 12.14.7.4.2]. In the case of point-to-point TESIs, its value is the ESP-VID of the MA's ESP that has the MEP's MAC address in its ESP-DA field.

21.7.5.3 Reverse MAC

This Reverse MAC field refers to the ESP-SA of any of the point-to-point ESPs of the MA, if the LBM/LTM is initiated on a root PBB-TE MEP or to the ESP-DA of the point-to-multipoint ESP of the MA, if the LBM/LTM is initiated on any of the leaf MEPs.

21.8 Linktrace Message Format

The format of the LTM is shown in Table 21-23.

Table 21-23—Linktrace Message format

Field	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 21-24.

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.

Table 21-24—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

21.8.3 LTM Transaction Identifier

(4 octets) A MEP copies the contents of the nextLTMtransID variable (20.41.1) 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 following TLV shall be included in every LTM transmitted by a MEP configured on a PBB-TE MA. The TLV will not be present in any other type of MA.

- d) PBB-TE MIP TLV (21.7.5).

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 an EUI-48 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 21-25.

Table 21-25—LTM Egress Identifier TLV format

Field	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 21-26.

Table 21-26—Linktrace Reply format

Field	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 21-27.

Table 21-27—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 21-28.

Table 21-28—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 20.47.1.2].	2
RlyMPDB	The Egress Port was determined by consulting the MIP CCM Database [item b) in 20.47.1.2].	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 21-29.

Table 21-29—LTR Egress Identifier TLV format

Field	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 21-30. 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 21-30—Reply Ingress TLV format

Field	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 either is 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 21-31.

Table 21-31—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 21-32. 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.

Table 21-32—Reply Egress TLV format

Field	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 value in the Egress Action field is one of the values enumerated in 21.9.9.1.

Validation Test: The Egress MAC Address field contains an Individual, and not a Group, MAC address.

Validation Test: The Length field either is 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 21-33.

Table 21-33—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.

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 (6.17) 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. If multiple Maintenance Associations, monitoring VLAN service instances, are configured on a single Bridge Port, the *vlan_identifier* parameter of the EISS and the MD Level in the CFM PDUs are used to multiplex CFM PDUs for the various Maintenance Associations. Non-CFM data frames for these service instances 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. MAs that monitor backbone or TESIs use I-SIDs or a combination of destination addresses, source addressees, and ESP-VIDs, respectively as service distinguishing parameters, instead of VIDs.

22.1.1 Preliminary positioning of Maintenance Points

Figure 22-1 illustrates how MPs are associated with particular VIDs or TESIs. The EISS Multiplex Entity (6.17) 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.

NOTE—For MPs monitoring backbone service instances, it is the Backbone Service Instance Multiplex Entity (6.18) that is used to distinguish MPs by I-SIDs and associate MPs with particular backbone service instances as illustrated in Figure 26-2.

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.

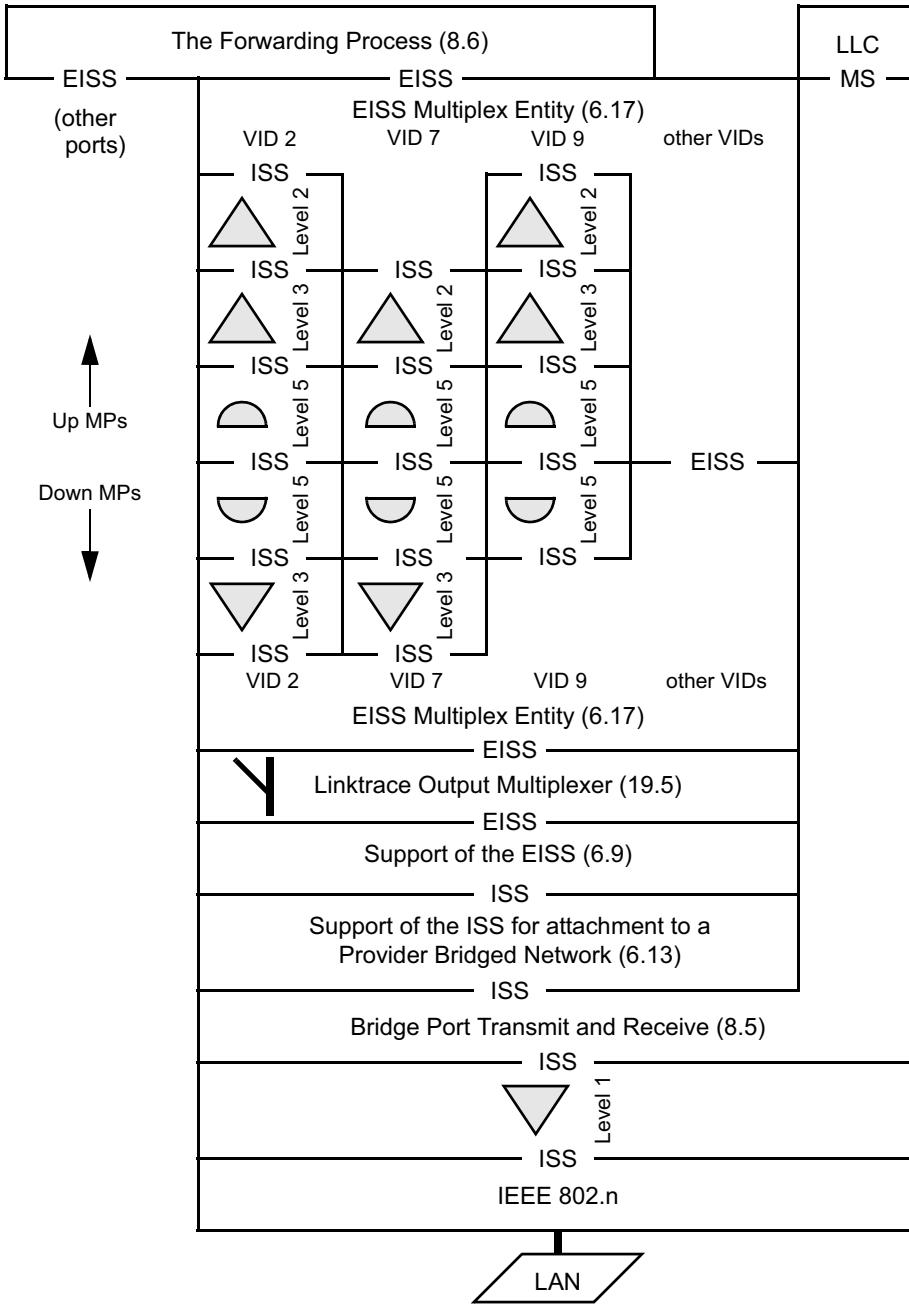


Figure 22-1—MEPs and MIPs distinguished by VID (incomplete picture)

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 VID they are associated with.

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 8-10 of IEEE Std 802.1Q-2005. We can notice two facts about Figure 8-10:

- a) With the exception of Frame filtering (8.6.3), every one of the functions in Figure 8-10 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 8-10 are exactly those of the EISS, and therefore each of the components in Figure 8-10 could be described as a shim.

From these facts, we can reconstruct Figure 8-10 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 (8.6.6, 8.6.7, and 8.6.8).

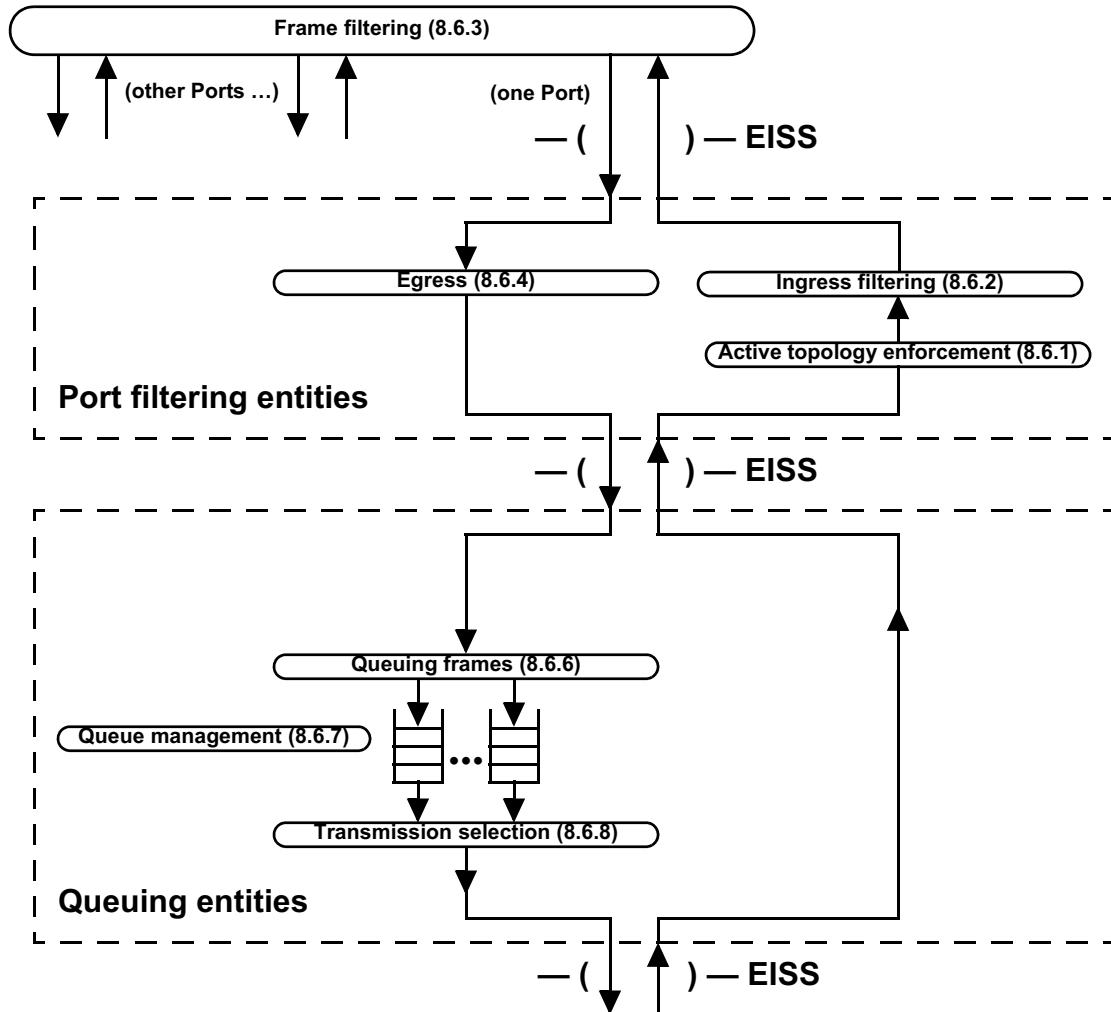


Figure 22-2—Alternate view of Forwarding process

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.

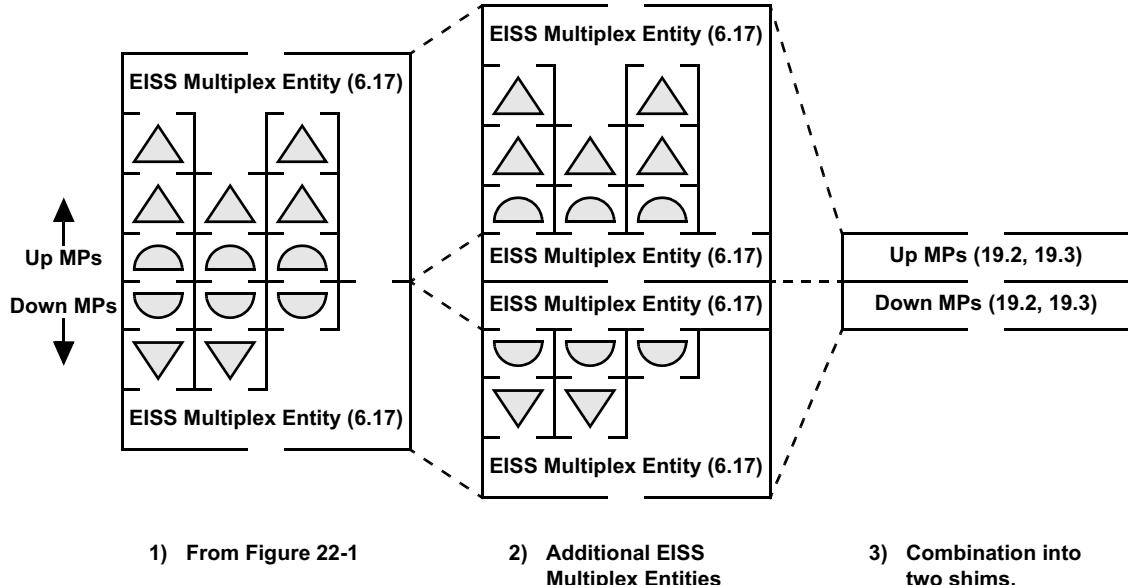


Figure 22-3—Combining per-VLAN Maintenance Points into two shims

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 filtering (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 VID 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.³³ 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.

³³The 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.

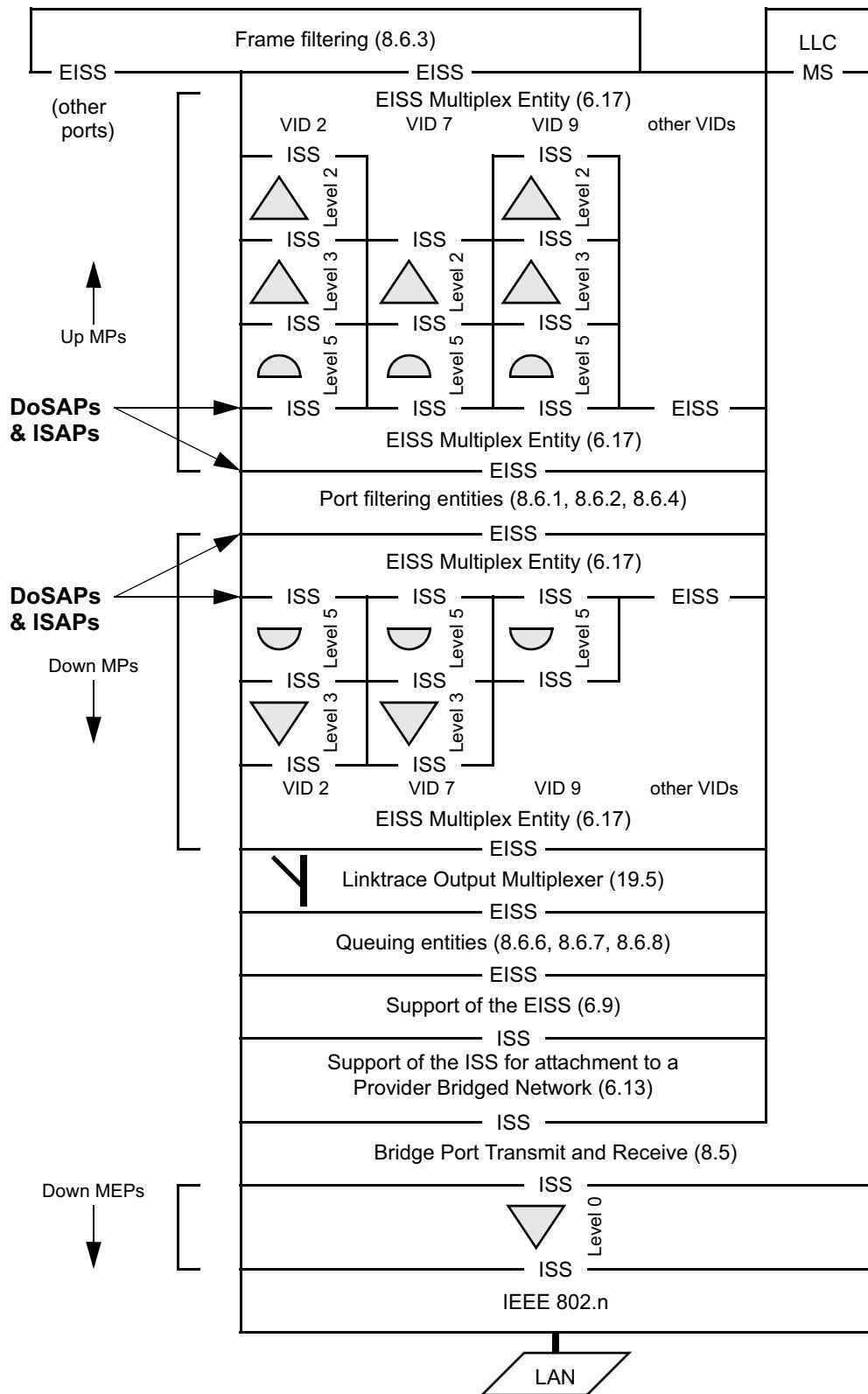


Figure 22-4—More complete picture of Maintenance Point placement in a Bridge Port

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 VID, there can be from zero to eight Up MEPs, ordered by increasing MD Level, from Frame filtering toward Port filtering. In the case of PBB-TE MAs, more than one Up MEP can share the VID and MD Level.
- 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 VID, if any. Backbone service instance MHFs use the Backbone Service Instance Multiplex Entity (6.18) and are placed as depicted in Figure 26-2.
- 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 (8.6.6, 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) No Down MEPs exist for PBB-TE MAs.
- e) An LOM is positioned below the Down MPs belonging to MAs that are attached to specific VIDs to handle forwarded LTMs.
- f) 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).
- g) Up to 8 Up MEPs, ordered from highest MD Level to lowest, belonging to backbone service instance MAs that are identified by the same I-SID, can be placed below the Bridge Port connectivity (8.5.1) using the Backbone Service Instance Multiplex Entity (6.18) as depicted in Figure 26-2. They are ordered by decreasing MD Level going away from the multiplexed SAP.
- h) Up to 8 Down MEPs, ordered from highest MD Level to lowest, belonging to backbone service instance MAs that are identified by the same I-SID, can be placed below the Bridge Port connectivity (8.5.1) using the Backbone Service Instance Multiplex Entity (6.18) as depicted in Figure 26-2. They are ordered by decreasing MD Level going away from the MAC Relay Entity.
- i) 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.
- j) 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 Port on a Bridge could, in theory, have MHFs for multiple service instances at different MD Levels. However, that would require a vector of 4094 MD Level registers instead of one register to identify the CFM PDUs 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.

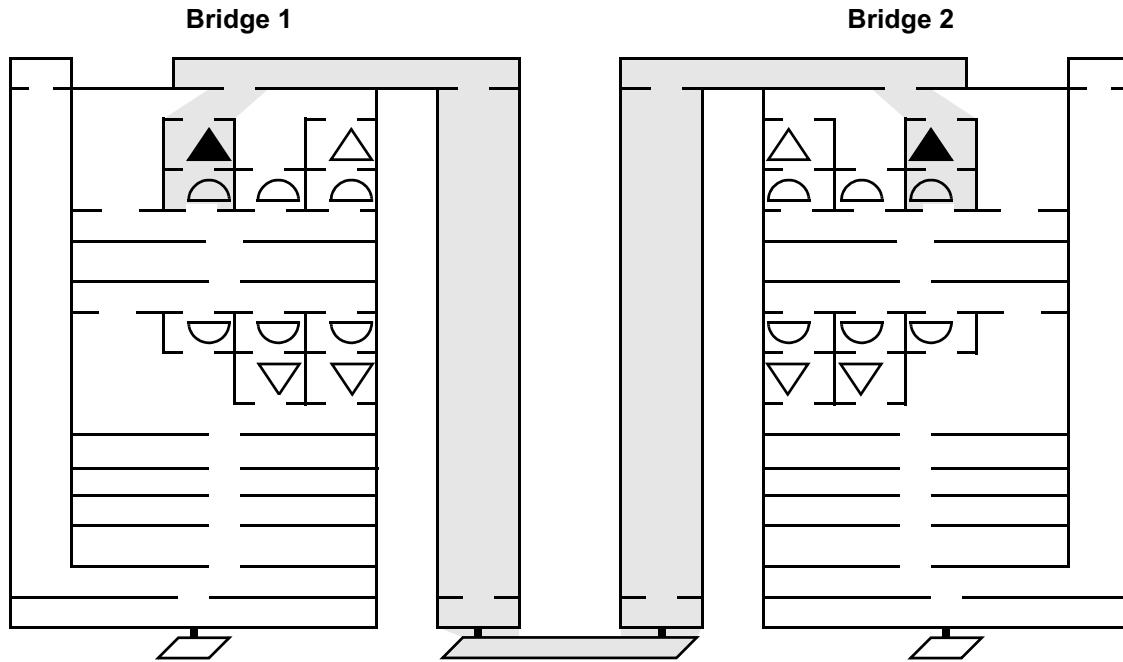


Figure 22-5—Service instance spanning two Bridges protected by Up MPs

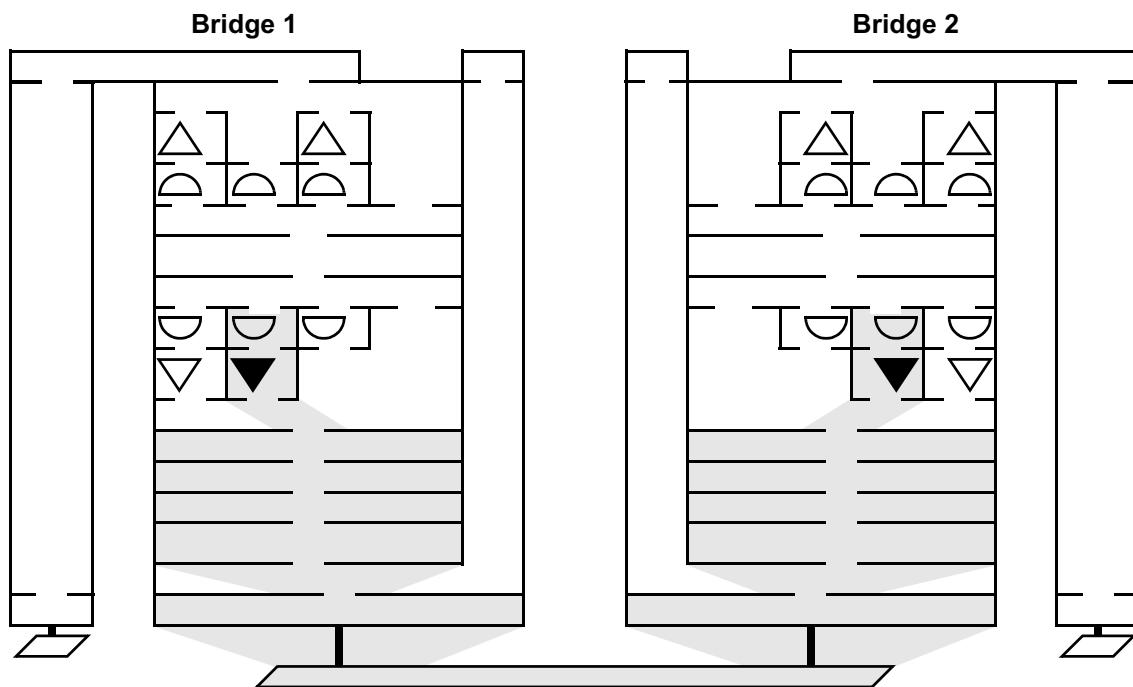


Figure 22-6—Service instance spanning two Bridges protected by Down MPs

22.1.5 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 VLAN service instance for a given frame. A VID-based MA or a PBB-TE MA with multiple VIDs does not require any more MIPs or MEPs than the corresponding MA with a single VID. All of the VIDs belonging to a given VID or TESI share the same MA. That is, one MAID is used for all CFM PDUs on the various VIDs in the same VID or TESI and Maintenance Domain. The choice of VID used to tag a given CFM PDU depends on the OpCode and the ESP, as specified in Clause 20.

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 VLAN untagged; the VLAN 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 or if it is associated with a backbone service instance only data with this instance as specified in (26.8); it does not separately protect the different VLANs.

Point c) above 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 VID 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 for VID identified service instances 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.

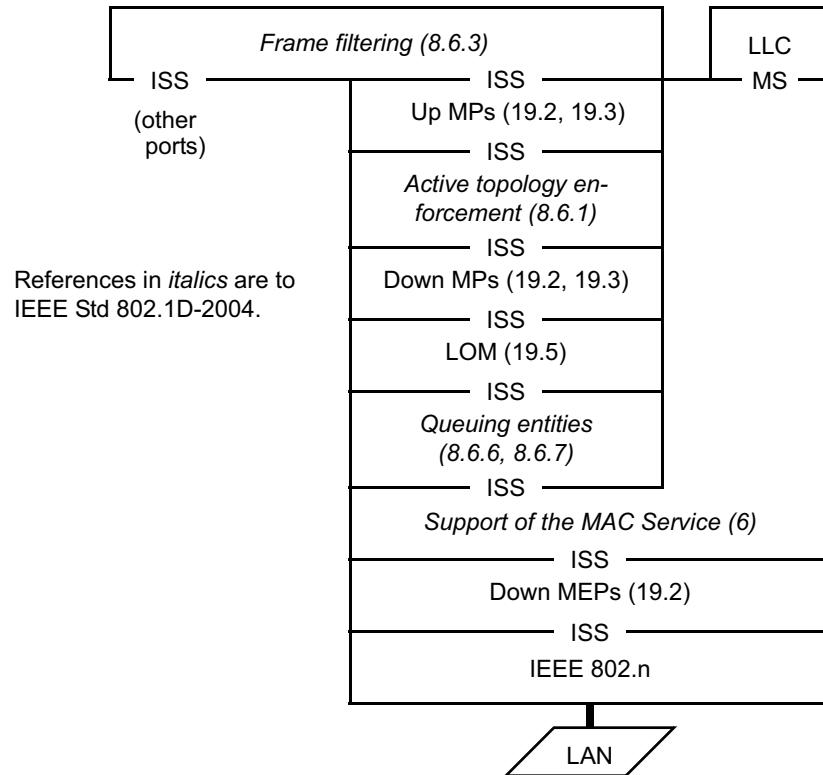


Figure 22-7—Maintenance Point placement in a non-VLAN aware Bridge Port

22.1.8 Maintenance Points and other standards

The relationship of Maintenance Points and the shims defined by two other standards, IEEE Std 802.1AX, Link Aggregation, and IEEE Std 802.1AE [B7], 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:

- 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.
- 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.
- 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.

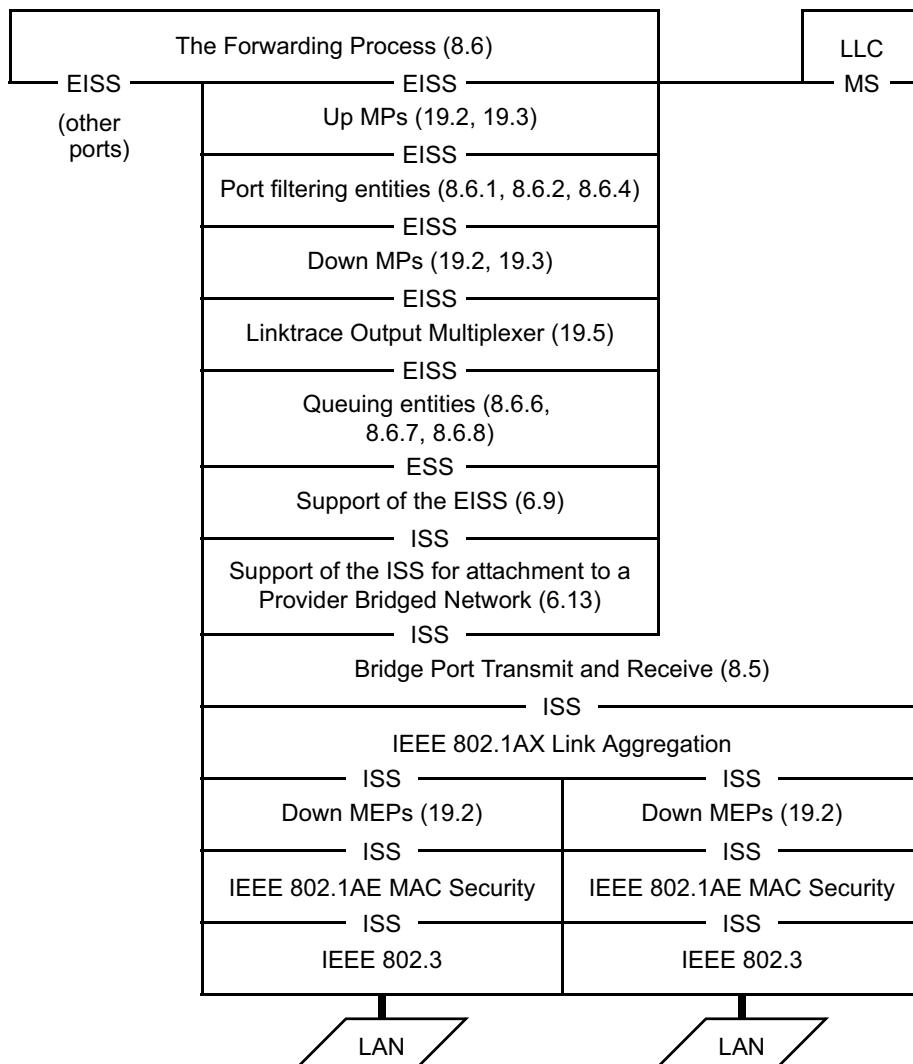


Figure 22-8—Maintenance Point placement relative to other standards

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 VID, and is thus above the Queuing entities, to protect the link.

When Link Aggregation is employed, Down MEPs attached to no VID 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 VID 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 toward a Down MEP attached to no VID; 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 VIDs 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 VID 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.

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 service instance can pass through a Bridge depends on:

- a) Enable Ingress Filtering per-Bridge Port parameter (8.6.2);
- b) Static Filtering Entries (8.8.1)
- c) Static VLAN Registration Entries (8.8.2);
- d) PIP configuration parameters (6.10);
- e) CBP configuration parameters (6.11);
- f) Dynamic VLAN Registration Entries (8.8.5); and
- g) 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 five 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. For example, a VLAN can pass through a Bridge unless there is a Static VLAN Registration Entry prohibiting its VID 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 four kinds of Maintenance Association managed objects: those that are attached to one or more specific VIDs in the Bridge, those that are attached to no VID, those that are attached to an I-SID, and those that are attached to a TESI.

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), I-SID, or TESI, and if so, which parameters [item b) in 12.14.6.1.3], including a specification of the MA's Primary VID [12.14.7.1.3:d];
- b) Whether the MEP is a Down MEP (pointing away from the MAC Relay Entity), or an Up MEP, (pointing toward 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, which shows a PNP or a CNP, 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. The third column, starting from the left, in the EISS Multiplex Entity depicted in Figure 22-9 corresponds to an ESP-VID. Only MIPs can be placed in this column for a PNP. MEPs can be created on a CBP and PIP as shown in Figure 26-2.

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

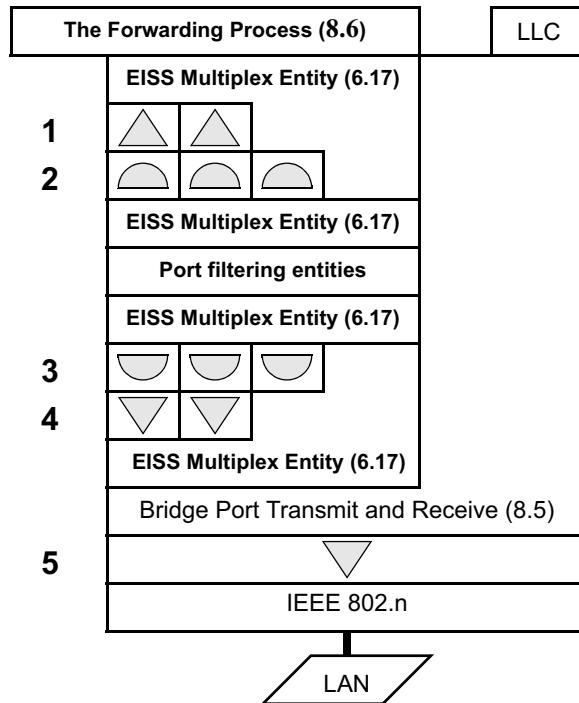


Figure 22-9—Creating MEPs and MIPs

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) or I-SID x on that port:

- a) The Bridge is initialized;
- b) A MEP is created or deleted on VID(s) or I-SID x on that Bridge Port;
- c) An Up MEP is created or deleted on VID(s) x , which is (are) not allocated to a TESI, on any Bridge Port or an Up MEP is created or deleted on I-SID x ;
- d) A change occurs in whether VID(s) or I-SID x 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) or I-SID x .

A MIP for VID-based service instances and TESIs is created by creating a pair of MHFs in positions 2 and 3 in Figure 22-9. MIPs associated with I-SIDs are created on the two sides of the multiplexed SAPs in the Backbone Service Instance Multiplex Entity as depicted in Figure 26-2. 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 service identifier x , which can be a list of VIDs, or an I-SID, the Maintenance Entity creates a list of active MD Levels. This list contains:

- g) MD Level of each of the MAs (if any) that monitors service x and has a MEP configured on Bridge Port p ;
- h) MD Level of each of the MAs (if any) that monitors a service 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 monitors service 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 service 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 service 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
defMFExplicit	False	No
	True	Yes
defMFDefault	—	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 monitors service x , and has at least one Up MEP configured on some Bridge Port in the Bridge [see item h];
- 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 service instance identified by the parameter x on a Bridge Port are:

- n) **defMHFnone:** No MHFs can be created for this VID(s) or I-SID (the default value).
- o) **defMHFdefault:** MHFs can be created for this VID(s) or I-SID on any Bridge Port through which the VID(s) or I-SID 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.
- p) **defMHFexplicit:** MHFs can be created for this VID(s) or I-SID only on Bridge Ports through which this VID(s) or I-SID can pass, and only if there is a MEP at the next lower active MD-level on the port.
- q) **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 or I-SID, 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 or I-SID, one or more of the VIDs or I-SID 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) or the I-SID also in MA x , does have a MEP configured on the Bridge Port;
- b) **ConflictingVIDs:** MA x is associated with a specific VID list or I-SID, 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) or I-SID 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 or I-SID at one MD Level, but a MEP is configured on another VID or I-SID 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 VIDs 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 VIDs 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 MAC Address Registration Entries in the Filtering Database. Minimizing the number of MD Levels, and concentrating the MD Levels used toward 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 LBMAs 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,³⁴ 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 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 VIDs, each with a MEP, are configured on the Bridge Port, again with the specified transmission interval and average number of MEPs per MA.

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.

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	1000	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%

³⁴First TLV Offset = 70, 12 for MAC addresses, 6 for Type and fixed header, 1 for End TLV, 20 for inter-frame gap and preamble.

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%

22.6 CFM in Provider Bridges

The S-VLAN component of a 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. Subclause 15.4 describes a C-tagged service interface, illustrated in Figure 15-4. Figure 22-10 recasts 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 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.

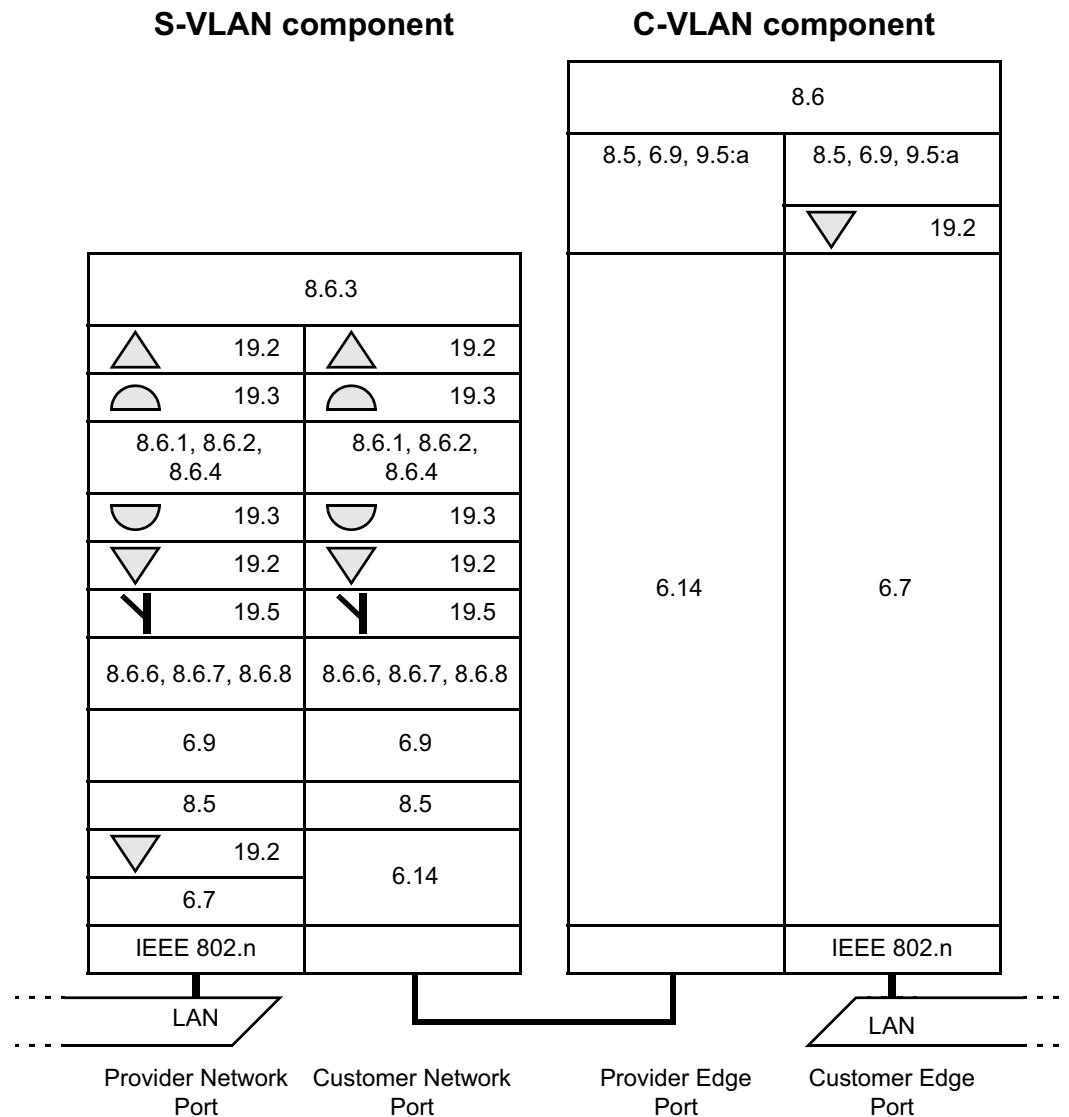


Figure 22-10—CFM in a Provider Edge Bridge C-tagged service interface

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:

- The extra MAs beyond the first MA would provide very little protection on a Port-based service;
- Each extra MA would add to the overhead of CFM PDUs; and
- 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.

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 (6.13)", 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.

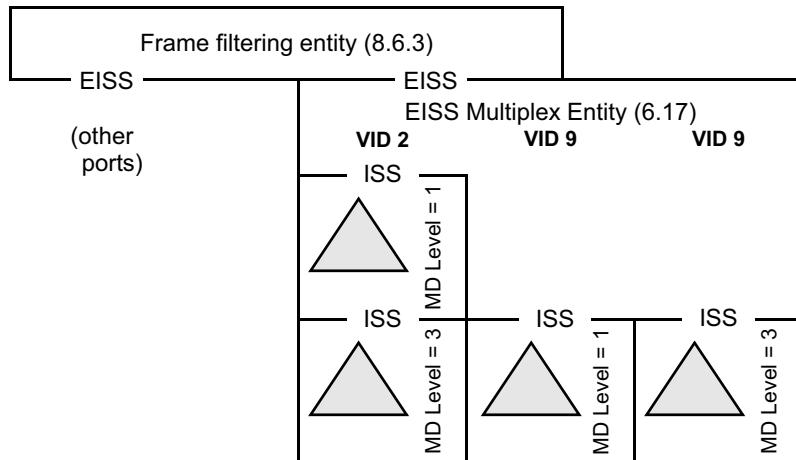


Figure 22-11—Up MEPs in a Management Port

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:

- a) A single Maintenance Domain can be configured at MD Level 0.
- b) An MA for each VID is configured in each Bridge.
- c) A single MEP for each VID 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 VID.

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.

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 MAC Address Registration Entry in the Bridge for the MD Level 3 group address shown in Table 8-13 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 MAC Address Registration Entries can be used in this manner that CCMs and LTMs are required to use the group addresses in Table 8-14 and Table 8-13.

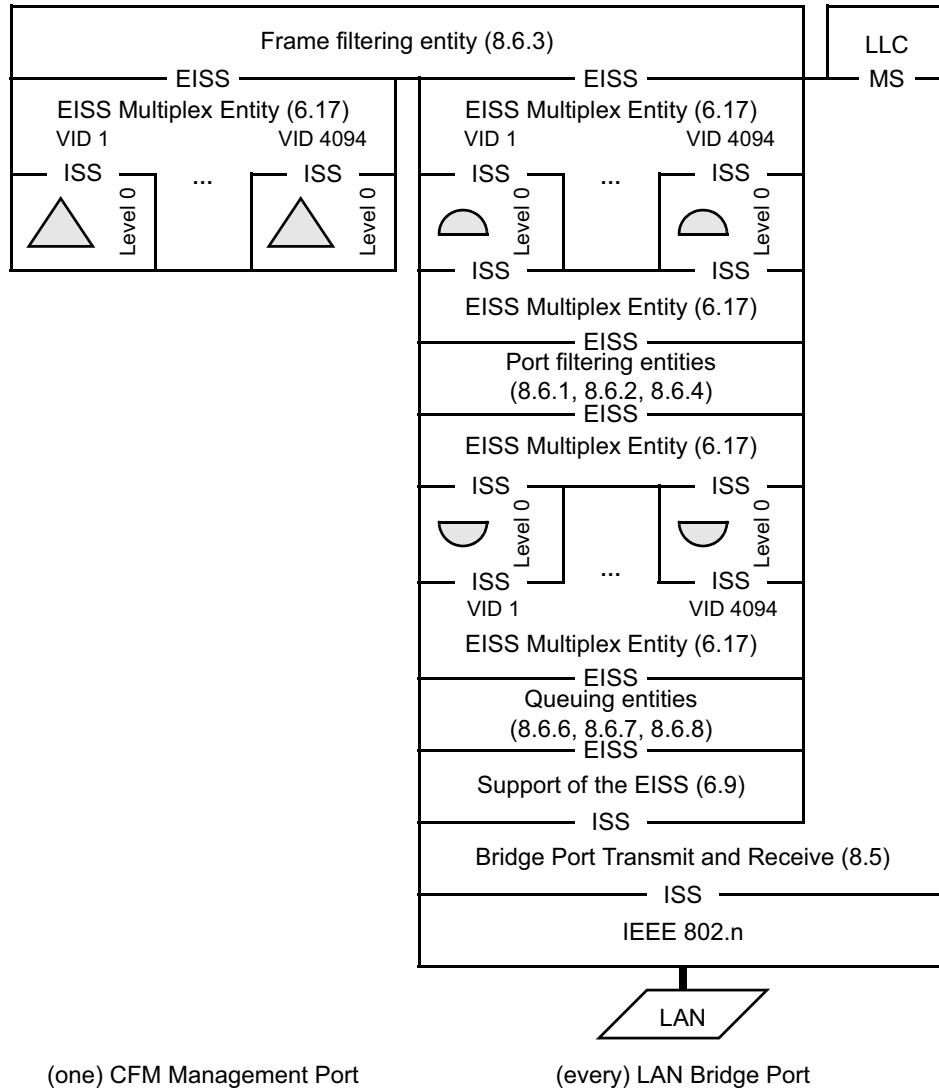


Figure 22-12—CFM in the enterprise environment

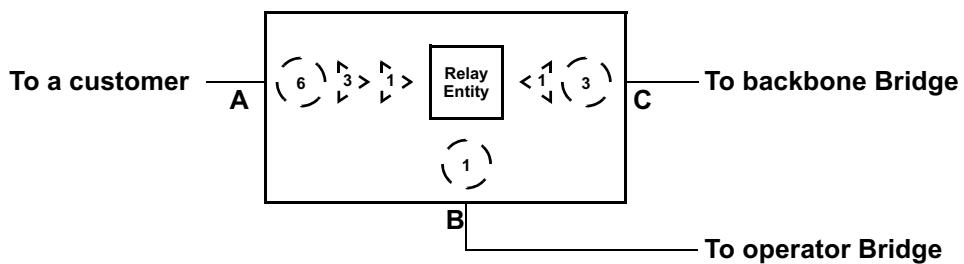


Figure 22-13—CFM on a Bridge conformant to IEEE Std 802.1Q-2005

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; MAC Address 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.

23. MAC status propagation

Individual LANs, each operating its own MAC and media access method specific procedures, can be connected by one or more TPMRs to form a composite LAN that is transparent to other bridges and stations and their configuration protocols. The MAC status protocol (MSP) ensures that changes in the connectivity provided by the composite LAN results in changes in the MAC_Operational status parameter (6.6.2) at each of the attached bridges and stations, just as if they were connected to an individual LAN.

NOTE 1—MSP is designed only for use with point-to-point LANs.

MAC_Operational provides rapid notification of connectivity failures and prompts the necessary initial protocol behavior to ensure that new connectivity has not caused an instantaneous data loop. If this MAC status parameter were not propagated by a TPMR, bridge protocols would have to rely on periodic transmissions to detect connectivity changes. On their own these periodic transmissions take longer to detect failures, and cannot detect the possibility of data loops until they have been created, as illustrated by the following examples.

Figure 23-1 shows a TPMR connecting two Bridge Ports.

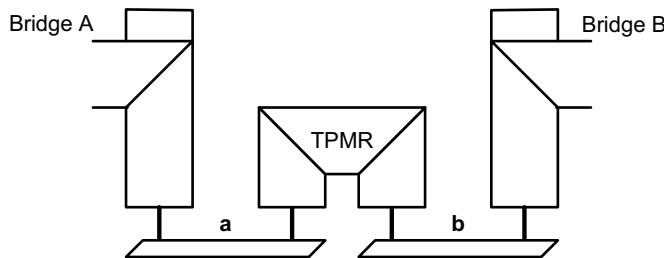


Figure 23-1—TPMR connecting two Bridge Ports

Assume that Bridge A is the spanning tree Designated Bridge for the LAN that comprises the individual LANs **a** and **b** and the TPMR, and that Bridge B's spanning tree Root Port is shown. If **a** fails and the TPMR does not propagate MAC_Operational, Bridge B will not reselect its Root Port until it has timed out the last BPDU from A.

Figure 23-2 shows a link that uses two TPMRs, perhaps because LAN **c** uses a non-802 technology together with an appropriate convergence function.

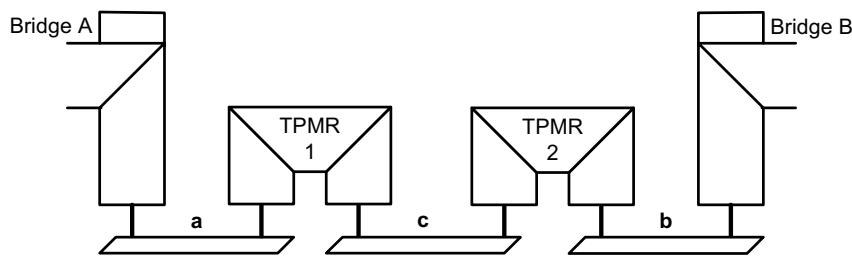


Figure 23-2—TPMR chain connecting Bridge Ports

Without MAC_Operational propagation, failure of **c** will not be immediately visible to either bridge. Worse, if **c** fails and is restored after a while, both A and B will believe themselves to be Designated Bridge for the composite LAN, and will forward frames until one receives a BPDU from the other, even if a data loop has been created.

When working correctly the MAC Service provides bi-directional connectivity or no connectivity at all. MAC_Operational is TRUE for each of two peers connected to the same LAN if they can communicate, and FALSE for either or both if they cannot. Protocols, such as the spanning tree protocols, that make use of MAC_Operational to detect new connectivity and initialize state machines rely on the last peer that sees MAC_Operational transition TRUE to enforce any necessary behavior after a connectivity change. For example, it is the last of two connected Bridge Ports to be powered on that enforces the necessary delay prior to setting operEdge TRUE (13.25).

NOTE 2—MAC_Operational being TRUE within a single station does not guarantee connectivity to any peer. Even if connected by a point-to-point LAN, the peer could have just reinitialized. It is not possible, given only the use of the LAN medium for communication, to arrange for two peers to transition MAC_Operational at exactly the same time.

This clause defines media independent propagation of MAC status between TPMRs and to attached stations, as follows. It

- a) Models MAC status propagation within the bridge architecture used to describe a TPMR (23.1)
- b) Provides an overview of the MAC Status Protocol (MSP, 23.2)
- c) Specifies state machines for MSP operation (23.3–23.9)
- d) Specifies the addressing, protocol identification, format, encoding, and validation of MAC Status Protocol Data Units (MSPDUs, 23.13–23.16)

The term *MAC status propagation* (3.91) describes the overall process of communicating a MAC_Operational parameter value through one or more TPMRs. MSP can use *link status notification*³⁵ (3.83) between some bridges and end stations, and *MAC status notification* (3.90) between others. These two notification methods differ, as follows:

- e) Link status notification uses frames to convey information about MAC_Operational. It does not interrupt or prevent other communication between adjacent bridges. However, it requires both the source and destination of the notification to implement additional protocol. Since it does not prevent communication it cannot, by itself, prevent loops caused by new connectivity.
- f) MAC status notification uses a layer management interaction with the local MAC Entity to change MAC_Operational for a peer user of the MAC service provided by an individual LAN. It is equivalent to bringing a link down, and is generally effective for MACs with specific point-to-point support. It also interrupts all other communication, including the use of in-band management to rectify an underlying problem.

Where management of a TPMR is permitted through one of its ports, the failure of an individual LAN not in the communication path does not cause MAC status propagation to prevent management connectivity. In Figure 23-2, for example, the failure of LAN **b** does not prevent connectivity to TPMR 2 from Bridge A. When **b** recovers, MAC_Operational for Bridge A's Port is “blipped,” i.e., taken FALSE for a brief period and allowed to return TRUE, thus meeting the requirement for transition when connectivity changes. While this slows the recognition of newly available connectivity, that is rarely an issue since several seconds hysteresis should be applied to any MAC_Operational status transition to prevent higher layer protocols “flapping.”

23.1 Model of operation

The model of operation presented in this clause allows the description of MSP functionality to be consistent with that of the general bridge architecture and operation (8.2, 8.3). Implementations of a TPMR may adopt any internal model of operation compatible with the externally visible behavior specified.

³⁵The use of the term link status notification in this standard is not to be confused with the term used in SNMP to refer to a type of trap notification.

The additional entities that model MSP operation, and their relationship to the other processes and entities that model the operation of a TPMR are illustrated in Figure 23-3. They comprise the following:

- A MAC Status Shim (MSS, 23.1.1) for each Port, that allows MSP to control the value of MAC_Operational presented to the TPMR Port connectivity function (8.5.2) and hence to the frame transmission and reception functions of the MAC Relay Entity, Higher Layer Entities, and MSPE.
- The MAC Status Propagation Entity (MSPE), that implements MSP, controlling each MSS to ensure that frames are not forwarded by the MAC Relay Entity until status propagation is complete, transmitting and receiving frames to support link status notification, and controlling the individual LAN's MAC to provide MAC status notification.

NOTE—In a TPMR the priority signaling functions shown in Figure 23-3 are provided by 6.20 Support of the ISS with signaled priority.

MSPDUs are sent to a standard group address. They are received by the MSPE, but are forwarded by the MAC Relay Entity (like any other frame) in order to communicate without a delay in each TPMR. The MSPE attaches to each Port by a TPMR Port connectivity function (8.5.2) that allows it to transmit to and receive from the attached individual LAN only, as shown in Figure 23-3.

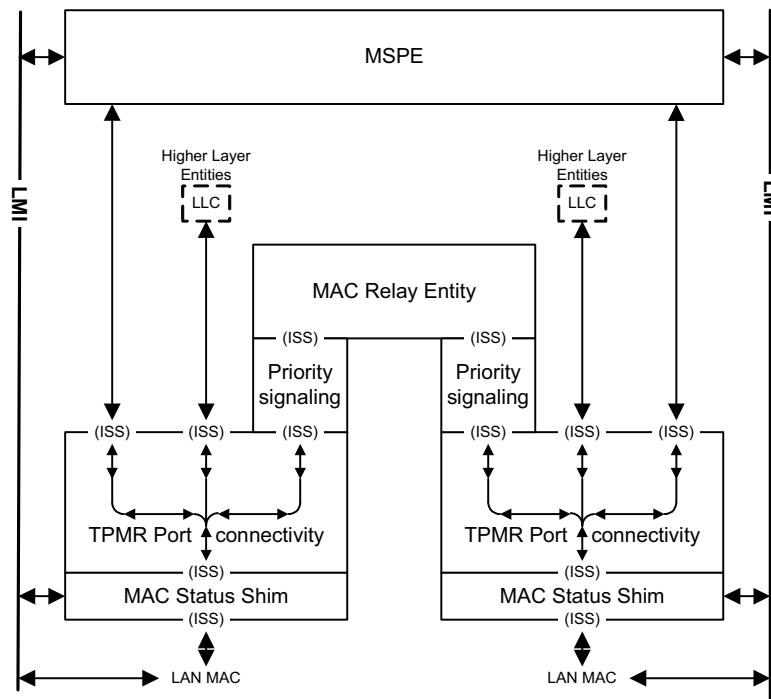


Figure 23-3—MAC Status Shims and the MAC Status Propagation Entity

23.1.1 MAC Status Shim

The MAC Status Shim (MSS) allows MSP to control the transmission and reception of frames through a Port by all the entities (including the MAC Relay Entity) supported by the TPMR Port Connectivity function (8.5.2). Each transmit request from the latter at the MSS's upper service access point (ISS-SAP) results in a corresponding transmit request at the lower ISS-SAP supported by the LAN MAC and each receive indication from the lower ISS-SAP results in a corresponding receive indication at the upper ISS-SAP, without omission or duplication, and with identical parameters, if and only if MAC_Operational for the upper ISS-SAP is TRUE.

The values of the MAC status parameters presented at the upper ISS service access point are determined as follows:

- **MAC_Enabled:** The value of this parameter is FALSE if the value of STM's disableMSS variable is TRUE. Otherwise, the value of this parameter is TRUE. STM communicates the value of its disableMSS variable using an LMI.
- **MAC_Operational:** The value of this parameter is TRUE if and only if the MSS's MAC_Enabled parameter is TRUE and the value of the MAC_Operational parameter provided by the LAN MAC at the lower ISS service access point is TRUE.

The MSPE uses the LMI to control and receive status from each MSS and individual LAN MAC, allowing generic management requests and indications to be tailored to the requirements of different MACs, as well as providing a way for one MSS to propagate status to another (via the MSPE), even though MAC_Operational for the MSS's upper ISS service access point is FALSE.

23.1.2 Relationship of Connectivity Fault Management to the MAC Status Shim

MEPs may be placed above or below the MSS. A MEP placed in the lower position and partnered with another MEP so that the pair spans a chain of TPMRs would obscure the LAN MAC's MAC_Operational parameter and interfere with the operation of MSP. For this reason, MEPs that protect individual LANs should be placed below MSS, and MEPs that protect chains of TPMRs should be placed above MSS.

23.2 MAC status protocol (MSP) overview

This clause provides examples of MSP operation as time sequence diagrams that show the following:

- a) Exchange of MSPDUs (MAC Status Protocol Data Units) transmitted and received to support link status notification.
- b) The values of MAC_Operational and MAC_Enabled provided by the following:
 - 1) Each MAC Status Shim (MSS), at its upper ISS service access point, to the TPMR Port connectivity function.
 - 2) Each LAN MAC to the MSS, at the latter's lower ISS service access point.

The value of MAC_Operational provided by the MSS is TRUE if, and only if the MSS' MAC_Enabled is TRUE and the value of MAC_Operational provided to the MSS by the LAN MAC is TRUE. The latter can be TRUE only if MAC_Enabled is TRUE for both the LAN MAC and its peer in the other station connected to the LAN. The MSPE can use the LMI to disable the MSS in order to prevent communication until MAC status propagation has occurred, and to disable the LAN MAC in order to provide MAC status notification to a peer service user. This clause uses the symbols defined in Table 23-1 to show combinations of the MAC status parameter values for each Port.

Throughout this protocol description, the term *LAN* is used to refer to individual LANs connecting adjacent TPMRs or a TPMR and an adjacent bridge. A *chain* is a series or part of a series of TPMRs connected by LANs, and a *link* is the connectivity provided by a chain between non-TPMR devices that communicate at a higher (sub) layer and perceive the entire link as a single transparent LAN.

The number of TPMRs in the following examples is deliberately high, four where one or two might be more common, in order to show all the necessary aspects of protocol behavior.

Signaling the addition of new or restored connectivity is considered first, as it is more difficult than signaling failure. Figure 23-4 shows the response to a new or recovered LAN connection in a link between two bridges that do not implement MSP—and thus require MAC status notification.

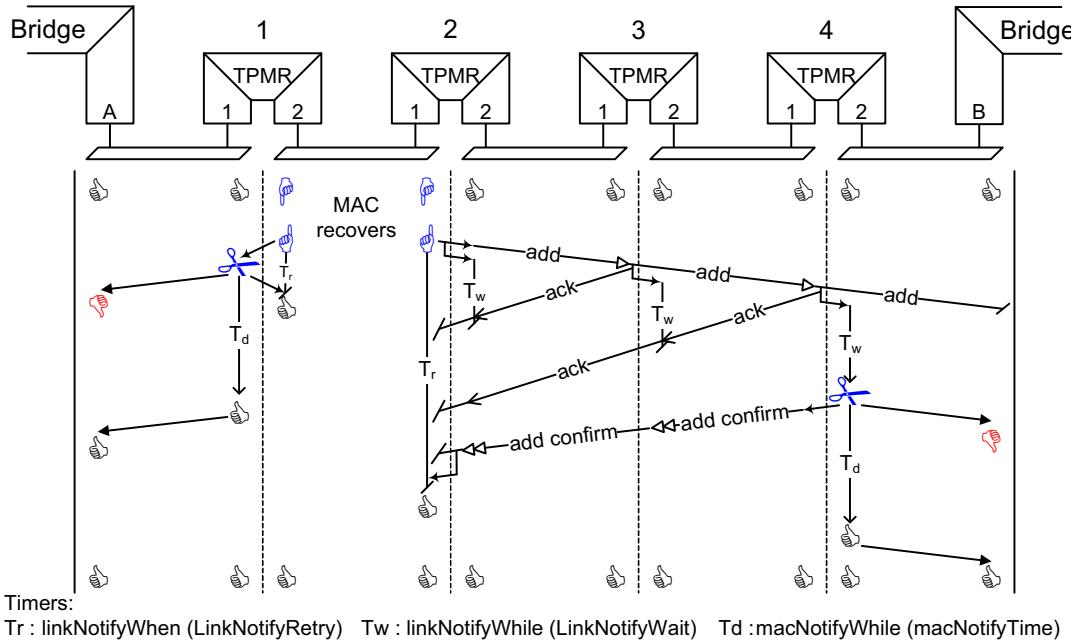


Figure 23-4—Adding connectivity

Table 23-1—Time sequence diagram symbols

		thumb up	thumb down	hand	pointing finger	scissors
MSS	MAC_Operational	T	F	F	F	F
	MAC_Enabled	T	T	F	F	—
LAN MAC	MAC_Operational	T	F	F	T	F
	MAC_Enabled	T	T	T	T	F

T = TRUE, F = FALSE, — = either TRUE or FALSE

Before the LAN that connects TPMR 1 to TPMR 2 is **MAC_Operational**, each MSPE disables its MSS (hand icon). This allows the MSPE to intercept the new connectivity as the LAN MAC asserts **MAC_Operational** (pointing finger icon). TPMR 2 propagates the new status to its other Port, transmitting an **add** MSPDU. TPMR 2 does not necessarily know that TPMR 3 implements MSP (or indeed is a TPMR) but uses the default MSP configuration—waiting before resorting to MAC status notification and starting a **linkNotifyWhile** timer (as do TPMRs 3 and 4) on receipt of the **add** (which is forwarded by the Relay Entity). Each TPMR responds to the **add** with an **ack** that clears the timer, but the bridge at the end of the chain does not (as it does not implement MSP). When TPMR 4's timer expires, its MSPE disables the LAN MAC (scissors icon), ensuring that **MAC_Operational** becomes FALSE (thumb down icon) for the bridge's Port, and starts a **macNotifyWhile** timer. When that timer expires, the MSPE reenables the LAN MAC so that the Bridge Port's **MAC_Operational** can become TRUE (thumb up icon) once more. As well as disabling the LAN MAC for its Port 2, TPMR 4 also transmitted an **add confirm** through Port 1, and receipt of this **add confirm** by TPMR 2 causes the latter to cancel its retry timer (**linkNotifyWhen**). The initial value of **macNotifyWhile** (**MacNotifyTime**) is sufficient for the **add confirm** to reach the TPMR 2, and for that TPMR's MSPE to enable the MSS, thus ensuring that there is connectivity.

between the bridges connected by the link when they have both reported MAC_Operational TRUE to their local protocol clients.

In Figure 23-4 the MSP configuration of TPMR 2's Port 2 differs from that for Port 1 of TPMR 1. The latter is explicitly configured to use MAC status notification immediately without waiting to see if its peer implements MSP.

Figure 23-5 illustrates the operation of MSP when the connectivity provided by an individual LAN is lost. The MSPE for each of the TPMR ports directly attached to the failed LAN begins by disabling the MSS for that Port (P) to ensure that connectivity does not ‘flap’. Otherwise the protocol proceeds as for connectivity addition, except that *loss* and *loss confirm* are used instead of *add* and *add confirm*, and the MSS' attached to the failed LAN are left disabled. The final state of the link components is the initial state assumed in Figure 23-4—both attached bridges have seen MAC_Operational transition to indicate that the connectivity has changed, and the TPMRs in each of the two chains can be reached and managed (if their individual configuration permits) through the attached bridge ports.

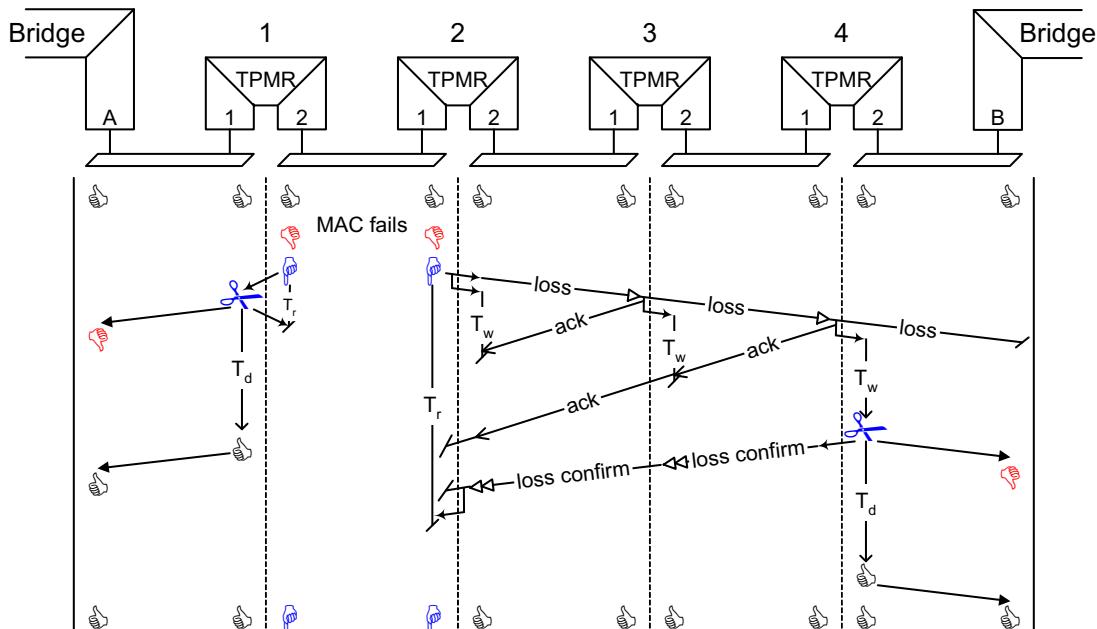


Figure 23-5—Losing connectivity

In the examples above, if an *add*, or *loss* is lost then the TPMR Port that transmitted or relayed that MSPDU will revert to using MAC status notification, as will a TPMR that fails to receive the *ack* from the next (or a subsequent) TPMR in the chain. If the MSPDU is lost on a LAN removed from the TPMR that initiated the link status notification, then that notification will be retried on expiry of the *linkNotifyWhen* timer, which also serves to protect against the loss of an *add confirm* or *loss confirm*.

Loss of a frame due to physical corruption is rare in LAN technologies, and frame losses due to buffer overrun are not expected when connectivity is being added (as the link is not usable prior to the addition) nor when a loss of connectivity is being signaled (as that connectivity loss will have prevented other frames from being added to the link). The loss of an MSPDU is most likely to be due to failure of one of the LANs, or to interruption of a TPMR's relay functionality by another MSS whose MSPE is also waiting for confirmation that a connectivity change that it has detected has been propagated to the end of the link. MSP does not, and cannot, ensure that information about each and every connectivity change reaches both ends of

the link. Unless link status notification or MAC status notification is disabled or the individual LANs fail to report MAC_Operational correctly, MSP does ensure that any change in connectivity is accompanied by one or more notifications at each end of the link, and that a continuous period during which MAC_Operational is reported TRUE at both ends of the link provides unchanged connectivity from 1 second after the start of the period to 1 second before the end. This guarantee meets the requirements implicit in the initial transmission and retransmission strategies of well designed protocols. Protocol clients of the MAC Service should not use the difference between *loss* and *add* MSPDUs to take different actions on receipt, though the distinction can be useful to a network administrator when investigating connectivity changes.

Figure 23-6 provides a common example of simultaneous change. TPMR 2 is powered on and initializes both its ports. When the LAN MAC becomes operational, each Port attempts to send a notification through the other, but cannot as they are both waiting for the connectivity addition to be confirmed. However, in this eventuality, each Port can provide the other with the confirmation required, because the peer system for each of the individual LANs will also have seen the initial MAC_Operational transition and will have initiated a notification toward its end of the link.

NOTE—The SNM state machine transition direct from SNM:LINK_NOTIFICATION to SNM:MAC NOTIFYING supports this behavior (see Figure 23-11).

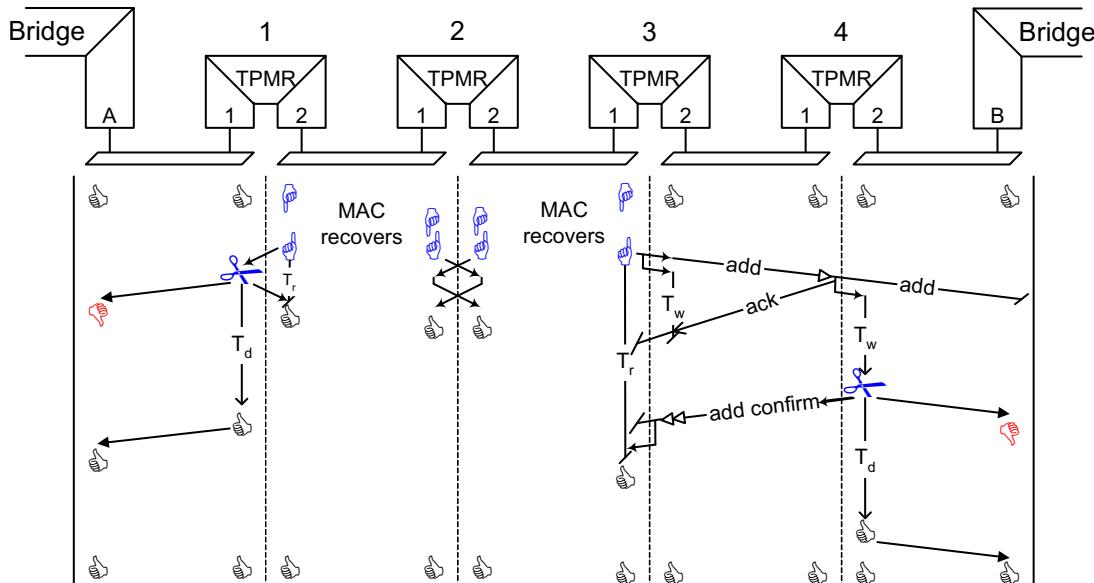


Figure 23-6—TPMR recovery

If the individual LAN that recovers (or loses) connectivity is at the end of the link, MSP ensures that the other end of the link is also aware of the status change as illustrated in Figure 23-7. This figure also shows the effect on MSP of including two nonstandard relays in the chain. Provided they (and their attached LANs) never fail, MSP can continue to operate as intended.

If TPMR 4 is configured to use MAC status notification immediately, it does not return an *ack* as the *add confirm* can be sent almost immediately (see Figure 23-8).

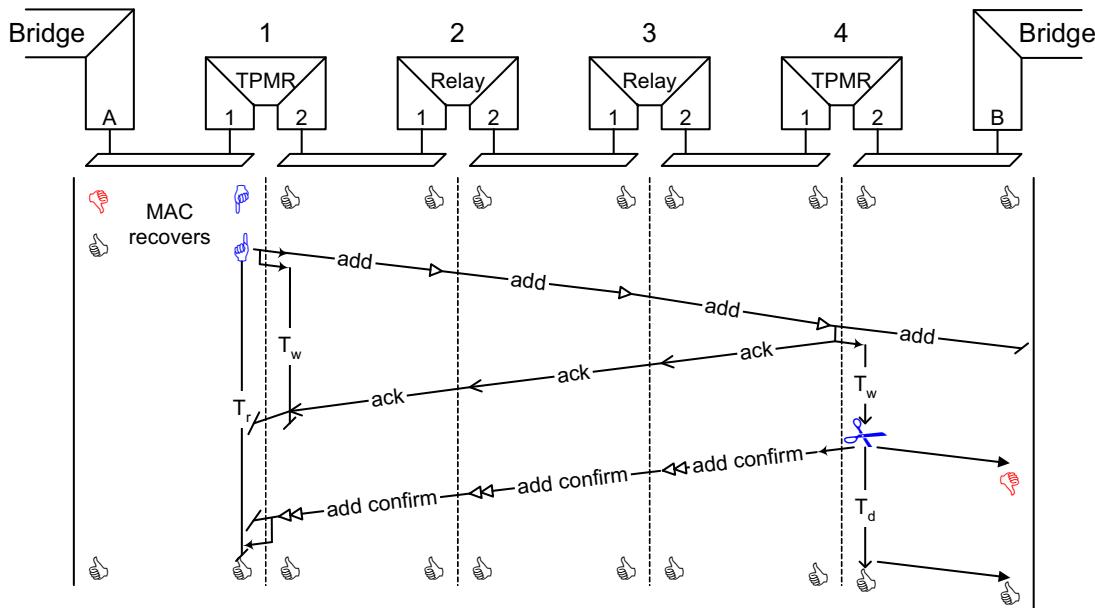


Figure 23-7—Notification from one end of the link to the other

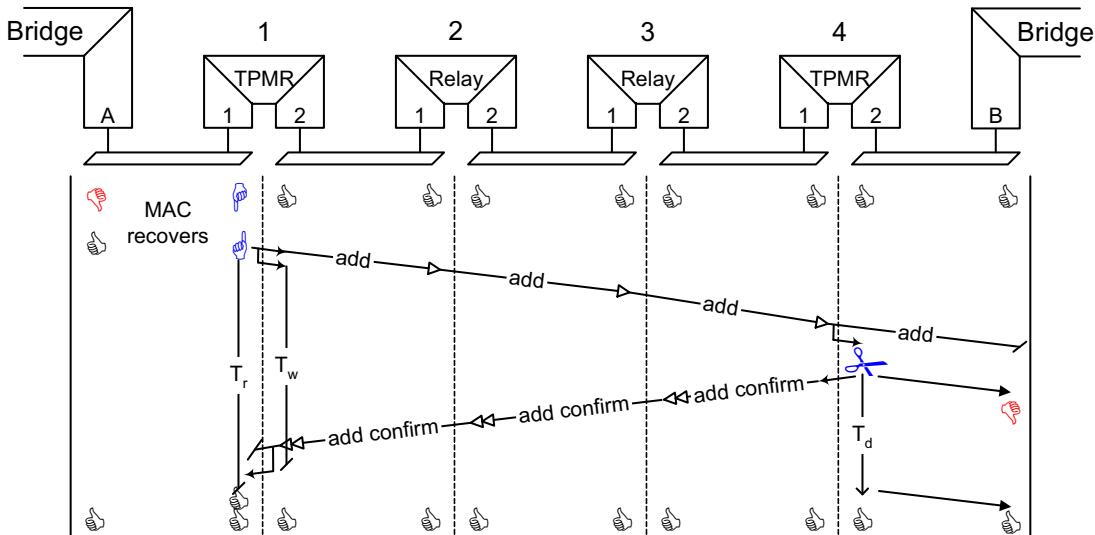


Figure 23-8—Immediate MAC status notification at the end of a link

23.3 MAC status protocol state machines

The operation of the MAC Status Propagation Entity (MSPE) is represented by an instance of each of the following for each Port of the TPMR:

- A Status Transition state machine (STM, 23.8)
- A Status Notification state machine (SNM, 23.9)
- A Receive Process (23.10)
- A Transmit Process (23.11)

Figure 23-9 shows the state machine variables that are used to communicate between these machines and processes, and that support management control over their operation. Variables prefixed with ‘r.’ are those of the corresponding state machines of the other Port of the TPMR.

The notational conventions used in Figure 23-9 and in the specification of the state machine are identical to those used in the specification of MSTP (Clause 13) and RSTP and are defined in Annex E.

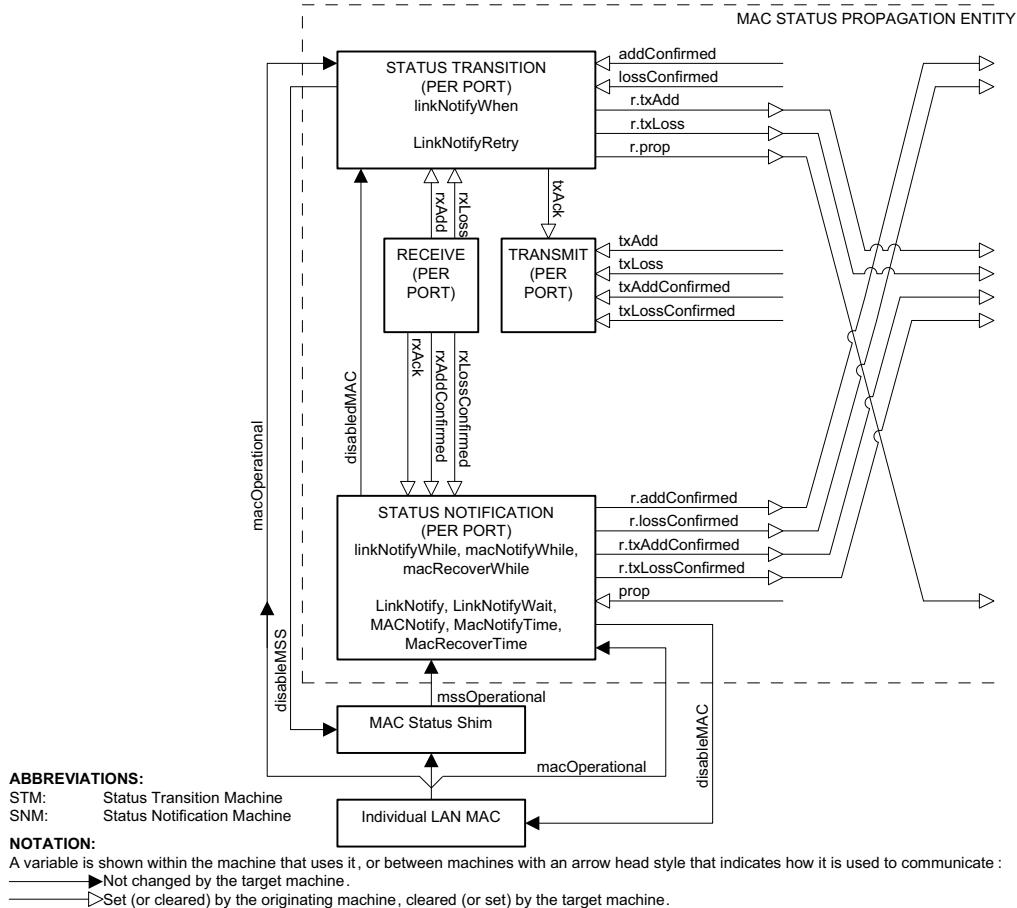


Figure 23-9—MSPE state machine overview

Port 1’s STM monitors MAC_Operational transitions for its own LAN, and tells Port 2’s Transmit process to send *add* or *loss* link notifications and Port 2’s SNM to monitor the progress of status notification, using MAC status notification if necessary. Port 2’s SNM receives the *ack* that indicates that it should wait for link status notification to complete, and the *add confirm* or *loss confirm* that indicates that completion. Whether link status or MAC status notification is used, Port 2’s SNM tells Port 1’s STM when the notification has been confirmed, so the latter does not have to retry the notification. Port 1’s SNM provides the same service to Port 2’s STM.

A *loss* or *add* notification that is received from Port 1’s own LAN is handled by its STM, which propagates the notification to Port 2’s SNM (in the same way as a local MAC_Operational transition) while transmitting an *ack* on Port 1’s LAN. Similarly Port 2’s STM transmits an *ack* for a link status notification received on its own LAN, and propagates the notification to Port 1’s SNM.

23.4 State machine timers

Timers are implemented by variables that are decremented on each timer tick, with timer expiry occurring when they reach zero.

23.4.1 linkNotifyWhen

Causes a link status notification to be sent on each expiry until the original status transition is confirmed.

23.4.2 linkNotifyWhile

Started when a change is first propagated through the Port, on expiry allows MAC status notification.

23.4.3 macNotifyWhile

Sets the time for which the MAC is disabled for MAC status propagation.

23.4.4 macRecoverWhile

Sets the time for which the MAC is permitted to be nonoperational, after being disabled, before the link is reported as lost.

23.5 MSP performance parameters

These parameters are not modified by the operation of MSP but are treated as constants by the state machines. They can be managed independently for each TPMR Port—default values and permissible ranges are specified in Table 23-2.

Table 23-2—MSP performance parameters

Parameter	Recommended or default value	Permitted range
LinkNotify	TRUE	TRUE or FALSE
LinkNotifyWait	0.4 s	0.2–1.0 s
LinkNotifyRetry	1.0 s	0.1–1.0 s
MACNotify	TRUE	TRUE or FALSE
MACNotifyTime	0.2 s	0.01–0.5 s
MACRecoverTime	0.1 s	0.02–0.5 s

23.5.1 LinkNotify

TRUE if the Port uses link status notification to propagate MAC status, and will wait to allow link status notification to succeed before using MAC status notification.

NOTE—If LinkNotify is FALSE, the TPMR still forwards *loss* and *add* notifications transmitted by other TPMRs prior to using MAC status notification, but the TPMR will not originate link status notifications.

23.5.2 LinkNotifyWait

The initial value of the linkNotifyWhile timer.

23.5.3 LinkNotifyRetry

The initial value of the linkNotifyWhen timer of the other TPMR Port.

23.5.4 MACNotify

TRUE if the Port uses MAC status notification.

23.5.5 MACNotifyTime

The initial value of the macNotifyWhile timer.

23.5.6 MACRecoverTime

The initial value of the macRecoverWhile timer.

23.6 State machine variables

23.6.1 BEGIN

This is a Boolean variable controlled by the system initialization process. A value of TRUE causes all TPMR state machines to continuously execute their initial state. A value of FALSE allows all state machines to perform transitions out of their initial state, in accordance with the relevant state machine definitions.

23.6.2 addConfirmed

Set by the other Port's SNM to tell STM that the addition has been confirmed. Cleared by STM.

23.6.3 disableMAC

Set by SNM to instruct the individual LAN MAC (via an LMI) to disable itself in a way that will cause MAC_Operational to be FALSE for the peer user of the MAC Service provided by that LAN.

NOTE—The state machines do not assume that a client of the MAC can tell whether the MAC is not operational because that client has disabled it, or whether some other client has disabled it.

23.6.4 disabledMAC

Set by the SNM when it has set disableMAC and for MACRecoverTime after disableMAC has been reset, so that the STM does not conclude that a loss notification should be sent through the other TPMR Port.

23.6.5 disableMSS

Set (or reset) by the STM to instruct the MSS (via an LMI) to set its MAC_Enabled status parameter to FALSE (correspondingly, TRUE).

23.6.6 lossConfirmed

Similar to addConfirmed but confirms a loss.

23.6.7 macOperational

The value of MAC_Operational for the individual LAN MAC.

23.6.8 mssOperational

The value of MAC_Operational provided by the MSS to the TPMR Port's bridge transmit and receive function.

23.6.9 prop

Set by the other Port's STM to Add or Loss to notify SNM that a change is being propagated through the Port. Reset by SNM to None.

23.6.10 rxAck

Set by the Receive process to tell SNM that an acknowledgment has been received. Cleared by SNM.

23.6.11 rxAdd

Set by the Receive process to tell STM that an *add* notification is being propagated through the TPMR. Cleared by STM.

23.6.12 rxAddConfirm

Set by the Receive process to tell SNM that an *add confirm* has been received. Cleared by SNM.

23.6.13 rxLoss

Similar to rxAdd, but for a *loss*.

23.6.14 rxLossConfirm

Similar to rxAddConfirm, but for a *loss confirm*.

23.6.15 txAck

Set by the STM to instruct the Transmit process to send an acknowledgment. Cleared by the Transmit process.

23.6.16 txAdd

Set by the other Port's STM to cause transmission of an *add* notification. Cleared by the Transmit process.

23.6.17 txAddConfirm

Set by the other Port's SNM to cause transmission of an *add confirm* (through this Port) confirming that a received *add* message has been acted upon. Cleared by the Transmit process.

23.6.18 txLoss

Similar to a txAdd but causes a *loss* message to be transmitted.

23.6.19 txLossConfirm

Similar to a txAddConfirm but causes a *loss confirm* message to be transmitted.

23.7 State machine procedures

No procedures are defined beyond those represented in the state machines.

23.8 Status Transition state machine

The Status Transition state machine shall implement the function specified by the state diagram in Figure 23-10 and the attendant definitions contained in 23.4 through 23.7.

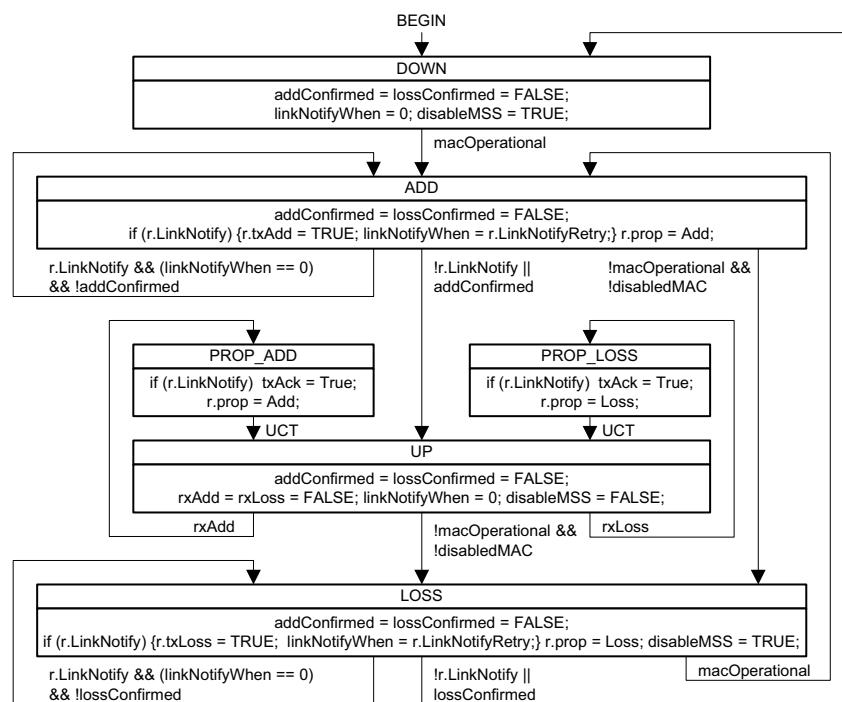


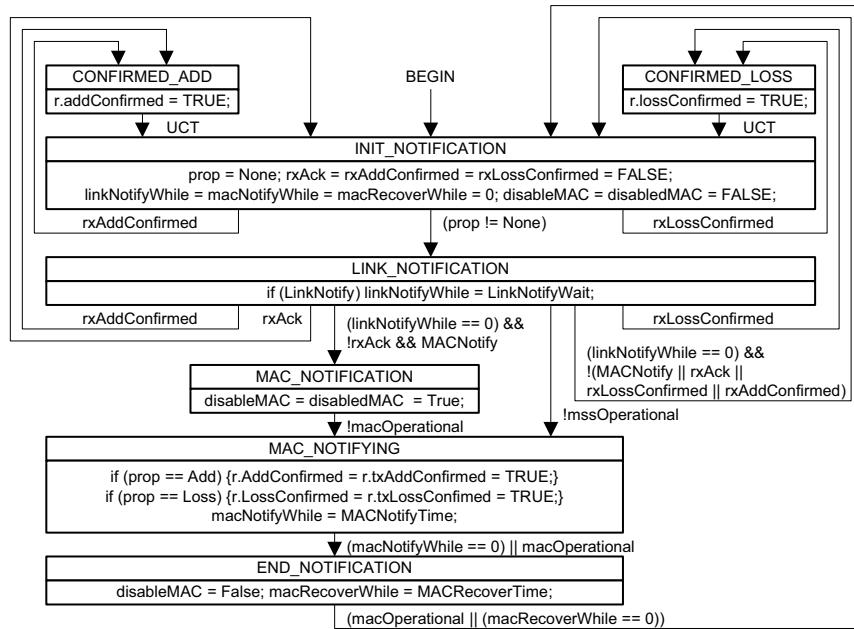
Figure 23-10—Status Transition state machine

23.9 Status Notification state machine

The Status Notification state machine shall implement the function specified by the state diagram in Figure 23-11 and the attendant definitions contained in 23.4 through 23.7.

23.10 Receive Process

The Receive Process shall receive and validate MSPDUs as specified in 23.16.

**Figure 23-11—Status Notification state machine**

23.11 Transmit Process

The Transmit Process shall transmit and encode MSPDUs as specified in 23.13–23.15. If the Transmit Process is instructed to transmit an MSPDU before it has had the opportunity to transmit a prior MSPDU, that prior MSPDU shall be discarded and not transmitted. If the **MAC_Operational** status provided by the MSS for the Port is FALSE, the MSPDU shall be discarded and not transmitted.

23.12 Management of MSP

An implementation of MSP in a TPMR:

- a) May allow the performance parameters (23.5) to be read by management.
- b) May allow the performance parameters (23.5) to be modified by management.
- c) May maintain each of the following counts for one or both ports of the TPMR:
 - **acksTransmitted**: The number of *acks* transmitted (23.6.15) by the Port's Transmit Process as a consequence of **txAck** being set.
 - **addNotificationsTransmitted**: The number of *adds* transmitted (23.6.16) by the Port's Transmit Process as a consequence of **txAdd** being set.
 - **addConfirmationsTransmitted**: The number of *add confirms* transmitted (23.6.17) by the Port's Transmit Process as a consequence of **txAddConfirm** being set.
 - **lossNotificationsTransmitted**: The number of *losses* transmitted (23.6.18) by the Port's Transmit Process as a consequence of **txLoss** being set.
 - **lossConfirmationsTransmitted**: The number of *loss confirms* transmitted (23.6.19) by the Port's Transmit Process as a consequence of **txLossConfirm** being set.
 - **acksReceived**: The number of *acks* received (23.6.10) by the Port's Transmit Process.
 - **addNotificationsReceived**: The number of *adds* received (23.6.11) by the Port's Receive Process.
 - **addConfirmationsReceived**: The number of *add confirms* received (23.6.12) by the Port's Receive Process.

- lossNotificationsReceived: The number of *losses* received (23.6.13) by the Port’s Receive Process.
- lossConfirmationsReceived: The number of *loss confirms* received (23.6.14) by the Port’s Receive Process.
- addEvents: The number of transitions to STM:ADD directly from STM:DOWN or STM:LOSS (23.8).
- lossEvents: The number of transitions to STM:LOSS directly from STM:UP or STM:ADD (23.8).
- macStatusNotifications: The number of transitions to SNM:MAC_NOTIFICATION (23.9).

23.13 MSPDU transmission, addressing, and protocol identification

MAC Status Protocol Data Units (MSPDUs) are transmitted and received using the service provided by an LLC entity that uses, in turn, a single instance of the MAC Service provided at an MSAP. In a TPMR the MSPE transmits and receives the MSPDUs, and the MSAP is provided by the TPMR Port connectivity function as illustrated in Figure 23-3. Each MSPDU is transmitted as a single MAC service request, and received as a single MAC service indication, with the following parameters:

- a) destination address (23.13.1)
- b) source address (23.13.2)
- c) MSDU
- d) priority (23.13.3)

The MSDU of each request and indication comprises a number of octets that provide EtherType protocol identification (23.13.4) followed by the MSPDU proper (23.15).

NOTE 1—For the purposes of this standard, the term “LLC entity” includes entities that support protocol discrimination using the EtherType field as specified in IEEE Std 802a-2003 [B6].

NOTE 2—The complete format of an MSP frame “on the wire” or “through the air” depends not only on the MSPDU format, as specified in this clause, but also on the media access method dependent procedures used to support the MAC Service.

23.13.1 Destination MAC Address

The destination address for each MAC service request used to transmit an MSPDU shall be the group address identified in Table 8-1 and Table 8-2 as “IEEE Std 802.1X PAE group address/Nearest non-TPMR Bridge group address”³⁶.

23.13.2 Source MAC Address

The source address for each MAC service request used to transmit an MSPDU shall be an individual address associated with the MSAP at which the request is made.

23.13.3 Priority

The priority associated with each MAC Service request should be the default associated with the MSAP. Transmitted MSPDUs are not Virtual LAN (VLAN) tagged or priority tagged.

³⁶This addresses was originally assigned by IEEE Std 802.1X-2001, and was identified as an S-VLAN component Reserved Address (Table 8-2) by the IEEE Std 802.1ad amendment to IEEE Std 802.1Q-2005. This standard further identifies it as the “nearest non-TPMR bridge group address,” to be filtered by all relay functions above the sub-layer specified for TPMRs (see Table 8-1 and Table 8-2).

23.13.4 EtherType use and encoding

All MSPDUs are identified by the EtherType specified in Table 23-3.

Table 23-3—MSP EtherType assignment

Assignment	Value
MAC Status Protocol EtherType	22-E2

Where an individual LAN MAC supports direct encoding of EtherTypes (as does IEEE Std 802.3, for example) the LLC entity shall encode the MSP EtherType as the first two octets of the MPDU. Otherwise (for IEEE Std 802.11, for example) the MSP EtherType shall be encoded in the initial octets of the MPDU according to the procedures specified in IEEE Std 802 for Subnetwork Access Protocols (SNAP).

NOTE—The SNAP discriminator comprises the octets AA-AA-03-00-00-00 prepended to the MSP EtherType.

23.14 Representation and encoding of octets

All MSPDUs consist of an integral number of octets, numbered starting from 1 and increasing in the order that they are put into a MAC frame. The bits in each octet are numbered from 1 to 8, where 1 is the low-order bit. When consecutive octets are used to encode a binary number, the lower numbered octet contains the more significant bits of the binary number.

When the encoding of (an element of) an MSPDU is represented using a diagram in this clause, the following representations are used:

- a) Octet 1 is shown toward the top of the page, higher numbered octets being toward the bottom.
- b) Where more than one octet appears on a given line, octets are shown with the lowest numbered octet to the left, higher numbered octets being to the right.
- c) Within an octet, bits are shown with bit 8 to the left and bit 1 to the right.

23.15 MSPDU structure

The MSPDU comprises the octets following the MSP EtherType. All MSPDUs comprise a Protocol Version (23.15.1) and a Packet Type (23.15.2).

Octet	
Protocol Version (23.15.1)	1
Packet Type (23.15.2)	2

Figure 23-12—MSPDU structure

23.15.1 Protocol Version

The MSP Protocol Version is encoded in all MSPDUs as a single octet, representing an unsigned binary number. Its value identifies the version of MSP supported by originator of the MSPDU. An implementation conforming to this specification shall encode the value 0000 0000 in this field. All other values are reserved.

NOTE—TPMRs that relay an MSPDU do not change its Protocol Version.

23.15.2 Packet Type

The MSP Packet Type is encoded as a single octet, representing an unsigned binary number. Table 23-4 lists the Packet Types specified by this standard, and the state machine variables set to indicate reception and transmission of MSPDUs of that type. All other possible values of the Packet Type field are reserved and shall not be used.

Table 23-4—MSP Packet Types

Packet Type	Value	Transmission	Reception
MSP-Add	0	txAdd (23.6.16)	rxAdd (23.6.11)
MSP-Loss	1	txLoss (23.6.18)	rxLoss (23.6.13)
MSP-Add Confirmed	2	txAddConfirm (23.6.17)	rxAddConfirm (23.6.12)
MSP-Loss Confirmed	3	txLossConfirm (23.6.19)	rxLossConfirm (23.6.14)
MSP-Ack	4	txAck (23.6.15)	rxAck (23.6.10)

23.16 Validation of received MSPDUs

To ensure that backward compatibility is maintained for future versions of this protocol, the validation and protocol version handling for all MSPDUs, follows general rules developed for this and other protocols. A received MSPDU shall be processed as specified by Table 23-4 if and only if

- a) The destination MAC address is the group address specified (23.13.1); and
- b) The MSPDU is identified by the MSP EtherType encoded as specified in 23.13.4; and
- c) The received MSPDU contains at least two octets, i.e. at least the Protocol Version and Packet Type; and
- d) The Packet Type is one of the values specified in Table 23-4.

Otherwise the received MSPDU shall be discarded. No other checks shall be applied to received MSPDUs, in particular the value of the Protocol Version is not checked and MSPDUs of length greater than the minimum of two octets are accepted as valid.

23.17 Other MSP participants

An end station or non-TPMR bridge attached to the end of a TPMR link can participate in link status notification, avoiding the need for the last TPMR in the chain to use MAC status notification and thus speeding the transition of the link to an operational state. Such a participant receives MSPDUs, but acts only on a received *loss* or *add*, notifying its protocol clients of the change in connectivity, and responding immediately by transmitting an *add confirm* or *loss confirm* as appropriate. There is no need for such a participant to transmit an *add*, *loss*, or *ack*, or act upon a received *ack*, *add confirm* or *loss confirm*, or to initiate MAC status notification.

24. <Reserved for a future amendment>

This clause has been reserved for use by a future amendment.

25. Support of the MAC Service by Provider Backbone Bridged Networks

A Provider Backbone Bridged Network (PBBN) comprises a set of Backbone Edge Bridges (BEBs) interconnected by some or all of the S-VLANs supported by a Provider Bridged Network (see Clause 16). Each BEB provides interfaces that encapsulate (or verify the encapsulation of) customer frames, thus allowing customer (C-MAC) addresses and VIDs to be independent of the backbone (B-MAC) addresses and VIDs administered by the PBBN operator and used to relay those frames across the backbone. The S-VLANs used to encapsulate customer frames are known as Backbone VLANs (B-VLANs), and the resources that support those VLANs are usually considered to be part of the PBBN.

NOTE 1—The term Customer used in the context of Provider Backbone Bridges may refer to a first Provider who is purchasing service from a second Provider. The first Provider deploys a PBN or a PBBN within its domain. The second Provider deploys a PBBN within its domain. In this case, the first Provider is identified as a Customer with respect to the second Provider. The term Customer used here may alternatively refer to each of a pair of Providers communicating as peers.

Figure 25-1 shows peer BEBs within the MAC sublayer and their relationship to customer and backbone bridging functions. The right-most BEB shown is modeled as comprising two types of VLAN-aware bridge components, an I-component and a B-component, connected together. The two BEBs on the left of the figure each comprise a single VLAN-aware bridge component of different types. The left-most BEB has a single I-component, while the BEB just to the right of the left-most BEB has a B-component. A T-component may also be used in place of an I-component.

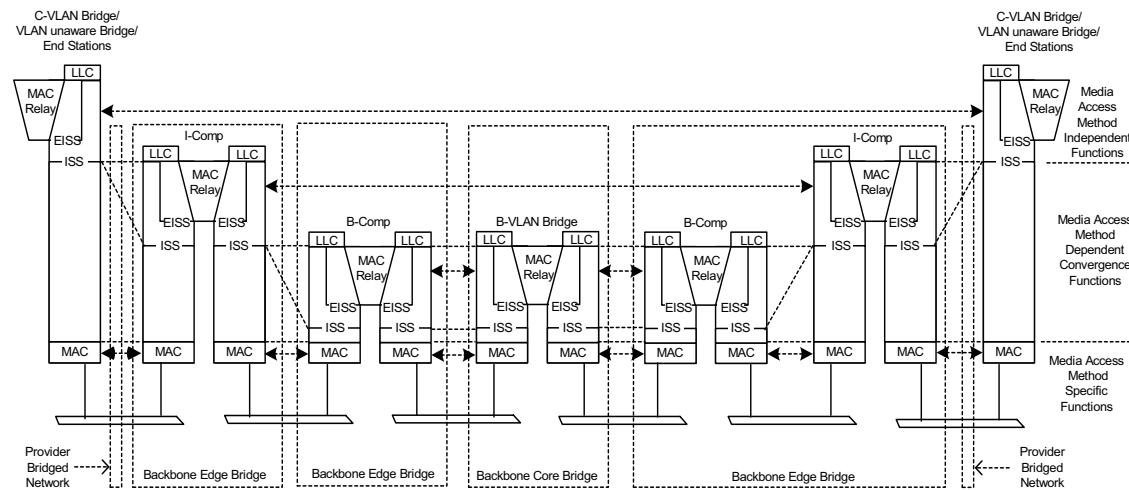


Figure 25-1—Internal organization of the MAC sublayer in a PBBN

Each I-component or T-component is responsible for encapsulating frames received from customers and assigning each frame to a backbone service instance. The backbone service instance consists of a set of BEBs that support a given customer's S-VLANs, and is uniquely identified within the PBBN by a Backbone Service Instance Identifier (I-SID). The customer frame is encapsulated by an I-TAG, which includes the I-SID, and a set of source and destination backbone MAC addresses. The backbone MAC addresses identify the BEBs of the backbone service instance where the customer frame will enter and exit the PBBN. If the I-component or T-component does not know which of the other BEBs provides connectivity to a given customer address, it uses a default encapsulating backbone MAC address that reaches all the other BEBs in the backbone service instance. Each I-component learns the association between customer source addresses received (encapsulated) from the backbone and the backbone source address, so subsequent frames to that address can be transmitted to the correct BEB.

A PBBN or a series of PBBNs providing the MAC service to attached end stations is typically modeled as a symmetric sequence of relay functions, as illustrated in Figure 25-1. The outermost peer relay functions are identified as I-components. The next peer relay functions in the sequence are identified as B-components. Between the peer B-components are one or more S-VLAN relay functions. A B-component relay forms the service layer to an I-component relay. A B-component relay forwards frames taking into account the identity of a B-VLAN (B-VID), while an I-component relay forwards frames taking into account the identity of an S-VLAN (S-VID).

A single B-component is responsible for relaying encapsulated customer frames to and from I-components and T-components, either within the same BEB or externally connected, checking that ingress/egress is permitted for frames with that I-SID, translating the I-SID (if necessary) and using it to assign the supporting connection parameters (backbone addresses if necessary and VIDs) for the PBBN, and relaying the frame to and from the Provider Network Port(s) (PNP) that provide connectivity to the other bridges within and attached to the backbone. A B-component performs the same functions when relaying frames to and from another B-component when two PBBNs interconnect (26.6.2).

The number of I-components (zero or more), T-components (zero or more), and B-components (zero or one) within a given BEB is a direct consequence of the type and number of external interfaces supported by that BEB. The I-components, T-components, and B-component may be in the same BEB or may be in different BEBs. The types of BEBs may be classified as I-BEB, B-BEB, T-BEB, and IB-BEB as depicted in Figure 25-2.

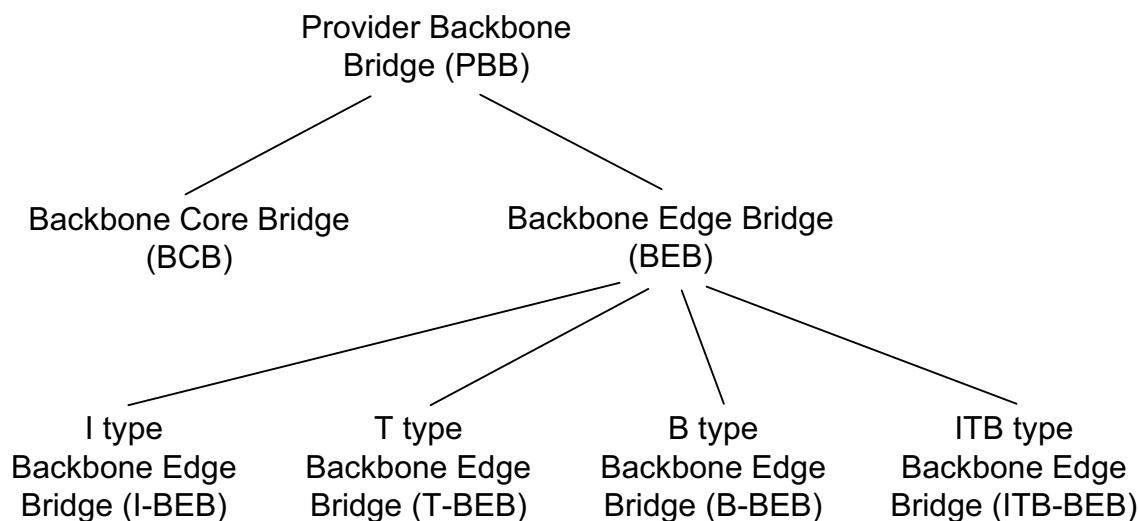


Figure 25-2—Provider Backbone Bridge terminology

This clause discusses the following aspects of PBBN service:

- a) Service transparency (25.1)
- b) Customer service interfaces (25.2, 25.3, 25.4, 25.5)
- c) Service instance segregation (25.6)
- d) Service instance and backbone service instance selection and identification (25.7)
- e) Service priority and drop eligibility selection (25.8)
- f) Service access protection (25.9)

NOTE 2—This standard makes use of the term *service* as defined by the OSI Reference Model (ISO 7498-1). In this sense, a service comprises a set of primitives and associated parameters, provided by one protocol layer in the architectural model to the protocol layer above, and the causal relationships between the primitives invoked by an upper

layer protocol entity in one system with those resulting indications to a peer entity in another system. The term service used by backbone providers, while including layering concepts, goes far beyond this formal definition, and commonly specifies some or all of the following: interfacing considerations across multiple protocol layers (including physical connectors, for example); selection of interface points; interfacing equipment; quality of service guarantees and measurement methods; charging methods and responsibilities; connectivity verification and other management tools; and regulatory issues.

25.1 Service transparency

The operation of PBBNs is, by design, largely transparent to Provider Bridges and Provider Bridged Networks as illustrated by Figure 25-1. The service provided by BEBs is transparent to the use of the MAC Service by end stations attached to the customer Bridged LANs through Provider Bridged Networks and transparent to the operation of media access method independent functions by Customer Bridges (CB).

The service is not transparent to the operation of media access method dependent convergence functions or to the operation of the media access method specific functions specified by standards for each media access method. Media access method dependent and specific functions operate between bridges—whether Customer Bridges, Provider Bridges, or Provider Backbone Bridges—attached to the same LAN.

25.2 Customer service interface

A backbone provider can offer to customers one or more types of service interfaces, each providing different capabilities for service selection, priority selection, and service access protection (25.7, 25.8, 25.9). In some cases it is assumed the customer provides an S-VLAN component of a Provider Bridge while in other cases, more generic customer systems are also allowed. There are four basic types of customer service interfaces—Port-based, S-tagged, I-tagged, and transparent service interface. The customer service interface types are summarized by Figure 25-3.

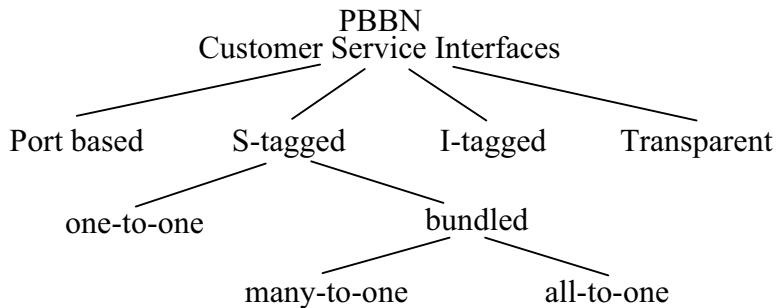


Figure 25-3—Customer service interface types

In all cases, segregation of different service instances is achieved at an interface wholly under the control of the backbone provider, and by verification of customer provided parameters that provide service instance selection. Stronger authentication and authorization of the attached customer systems may be achieved by use of IEEE 802.1X.

NOTE—The term service access protection describes provision of service access over multiple access LANs and (or) nodes with redundancy and rapid failover in case of failure of an access LAN or attached equipment.

Access to a given backbone service instance can be provided through different types of customer interfaces.

25.3 Port-based service interface

A PBBN may provide a Port-based service interface for customer attachment. The PBBN Port-based interface provides the same type of service to a customer as the PBN Port-based interface described in 15.3. A Port-based service interface is delivered on a Customer Network Port (CNP) provided by a BEB as illustrated in Figure 25-4 and Figure 25-5. A Port-based service interface may attach to a C-VLAN Bridge (5.9), IEEE 802.1D bridge, router, or end-station. The service provided by this interface forwards all frames without an S-TAG over the backbone on a single backbone service instance. All frames with an S-TAG that has a non-null VID are discarded by a Port-based service interface.

The Port-based service interface requires specific constraints on the configuration of the I-component. Each I-component CNP providing a Port-based interface is associated with one and only one VIP with exactly the same parameters for their tagging functions support. More specifically both CNP and VIP

- a) Have the Acceptable Frame Types parameter configured to Admit Only Untagged and Priority-tagged frames;
 - b) Have equal PVID values;
 - c) Are exclusive members of the same PVID member set;
 - d) Are members of the PVID untagged egress set.

Figure 25-4 illustrates a customer system attached to a PBBN over a Port-based service interface.

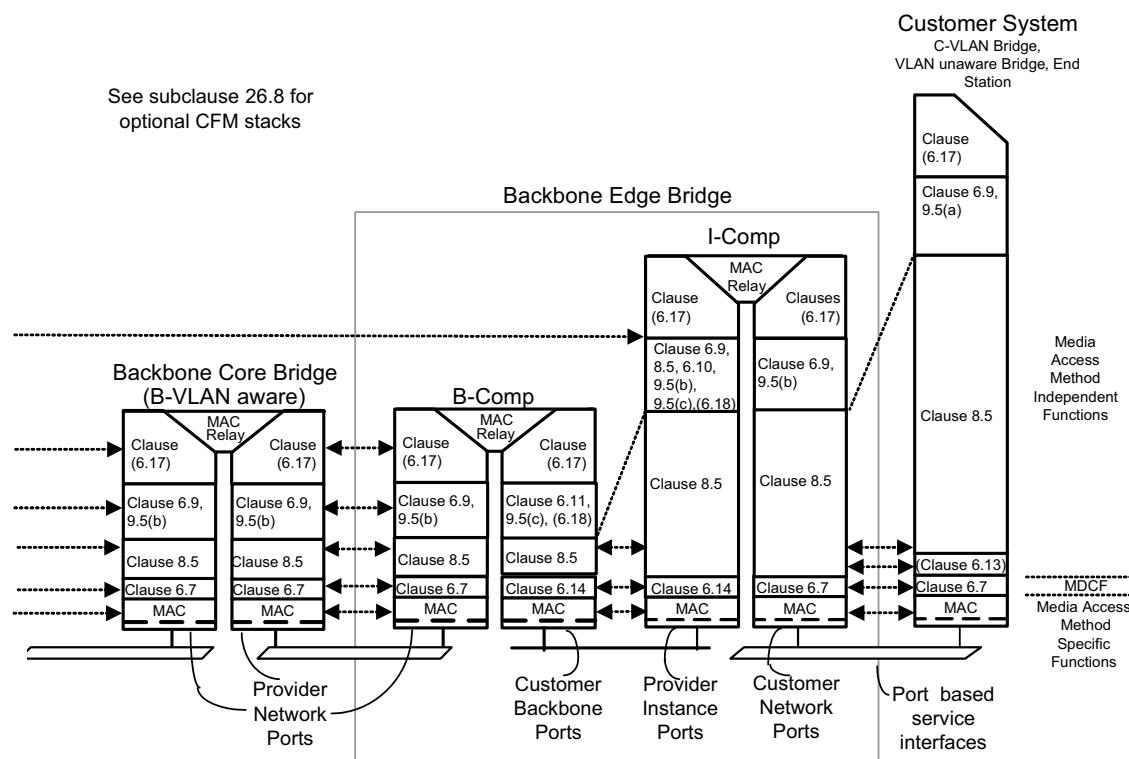


Figure 25-4—Port-based service interface

Figure 25-5 shows an example of equipment used to implement an unprotected Port-based service interface. For details on redundant connections and equipment, see 25.9. An I-component may support one or more Port-based service interfaces. Each CNP may be associated with one Port-based service interface. A CNP is connected to a customer system (or network) using a LAN.

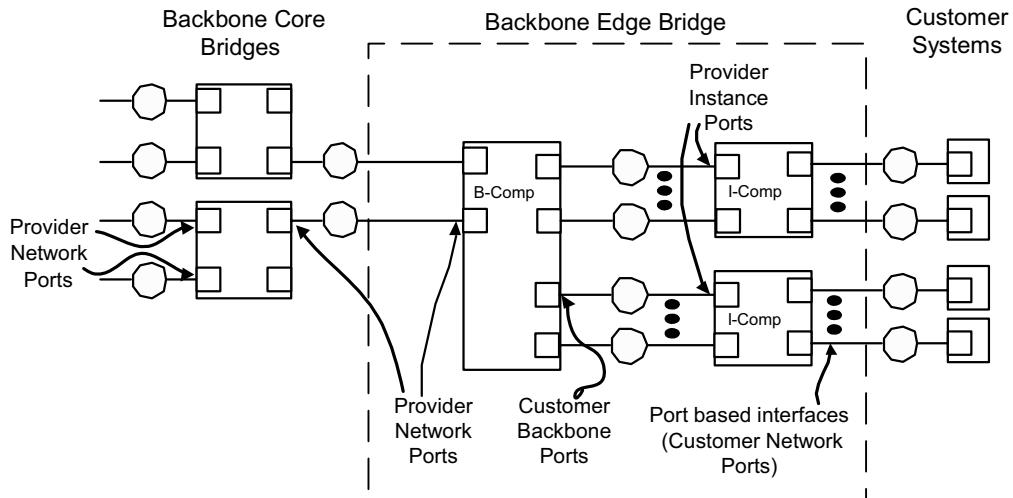


Figure 25-5—Port-based interface equipment

25.4 S-tagged service interface

The S-tagged service interface maps a service instance from a Provider Bridged Network, identified by an S-VID, to a backbone service instance on the PBBN, identified by an I-SID. There are two types of S-tagged service interfaces—one performing a one-to-one mapping of S-VIDs to I-SIDs and the other bundling S-VIDs to I-SIDs. Frames that are mapped to the I-SID are carried over the PBBN while frames that are not mapped to an I-SID are not carried over the PBBN.

NOTE 1—The restriction that each PBN S-VLAN map to a single backbone service instance on the PBBN allows the PBN equipment receiving frames to correctly identify the service instance used to deliver that frame and prevents the configuration of the I-component to create a multipoint service from point-to-point service instances, which could result in accidental creation of data loops. The backbone provider can offer a multipoint service through appropriate configuration of the B-VLAN component.

A PBBN may provide an S-tagged service interface for attachment to customer Provider Bridged Networks (15.5). An S-tagged service interface is provided by a BEB over a CNP as illustrated by Figure 25-7 and Figure 25-8. The attached Provider Bridges can in turn provide Port-based, C-tagged, or S-tagged service interfaces to their customers as described in 15.2, 15.3, 15.4, and 15.5.

The S-tagged service interface has the variations shown in Figure 25-3 under the S-tagged branch. The first variation, called a one-to-one S-tagged interface, uses a one-to-one mapping between S-VIDs and I-SIDs. This interface variation maps each S-VID to a single I-SID for use over the PBBN. The one-to-one mapped interface does not carry the S-TAG over the PBBN. The DEI and PCP bits may be regenerated on ingress and are then carried in the I-DEI and I-PCP bits in the I-TAG across the PBBN. On egress from the one-to-one S-tagged interface, the S-TAG can be deduced from the I-TAG received from the PBBN (the I-SID is mapped to an S-VID, the I-DEI and I-PCP bits may be regenerated and are then carried in the DEI and PCP bits).

The second S-tagged service interface variation is the bundling S-tagged service interface. This interface variation maps multiple S-VIDs to a single I-SID for delivery over the PBBN. To allow the remote end to reconstruct the S-VID, this interface variation will carry an S-TAG over the PBBN. On a bundled S-tagged interface, the DEI and PCP bits of the S-TAG may be regenerated and are then carried in both the DEI and PCP bits of the S-TAG and the I-DEI and I-PCP bits of the I-TAG over the PBBN.

A special case of the bundling S-tagged service interface is where all S-VIDs are mapped to a single I-SID. This special case is called an all-to-one bundling S-tagged service interface. The I-component used for an all-to-one bundled S-tagged service interface is constrained to supporting a single S-tagged service interface.

Figure 25-6 illustrates the information passed over each of the ISS interfaces of a BEB.

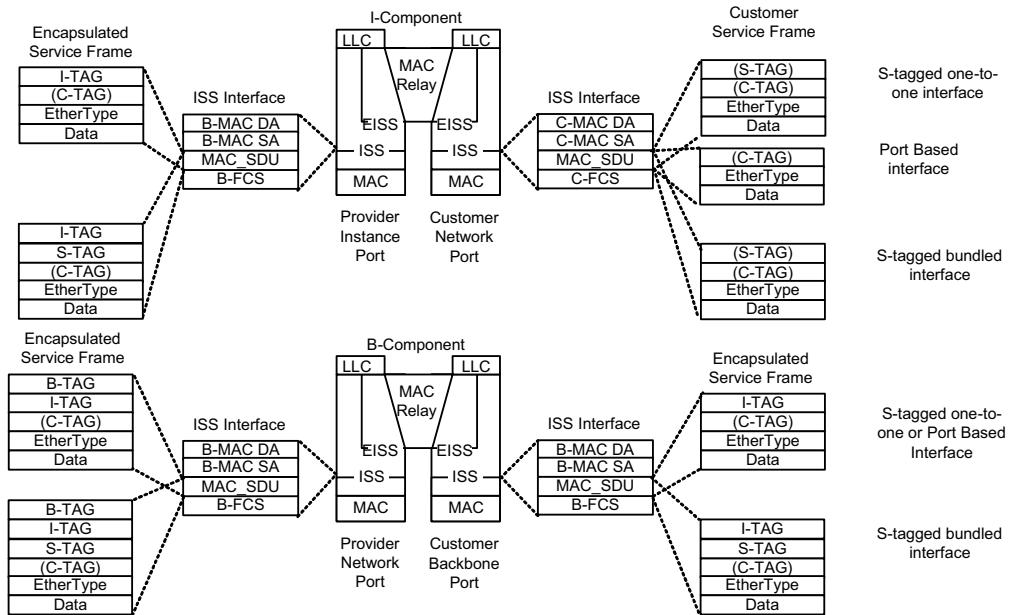


Figure 25-6—Encapsulated service frames at ISS

Figure 25-7 illustrates a customer network attached to a PBBN using an S-tagged service interface. The customer network uses Provider Bridges with S-VLAN components for connecting to the PBBN. The PBBN in turn is composed of BEBs interfacing to the customer Provider Bridges and Backbone Core Bridges used to forward frames between the BEBs.

See subclause 26.8 for optional CFM stacks

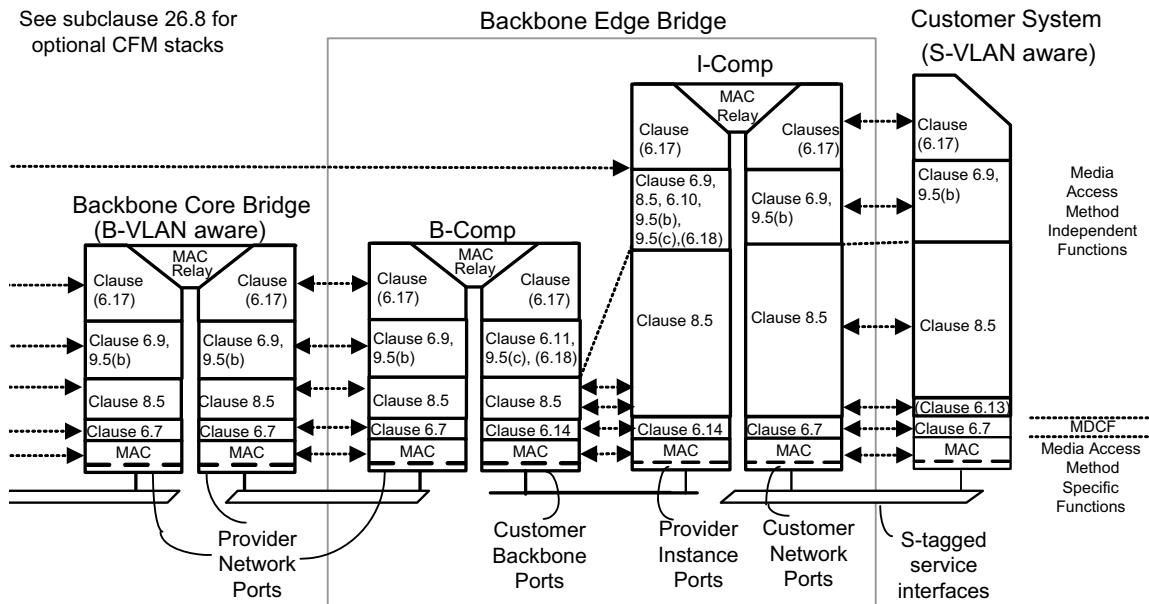


Figure 25-7—S-tagged service interface

Figure 25-8 shows an example of equipment used to implement an unprotected S-tagged service interface. For details on redundant connections and equipment, see 25.9. In this diagram, a BEB is formed by connecting two I-components and a B-component. These connections may be over a backplane or over a LAN. An I-component may support one or more S-tagged service interfaces. When an I-component supports an all-to-one bundled S-tagged service interface, the entire I-component must be dedicated to a single S-tagged service interface. Each CNP may be associated with one S-tagged service interface. A CNP is connected to a customer system (or network) using a LAN.

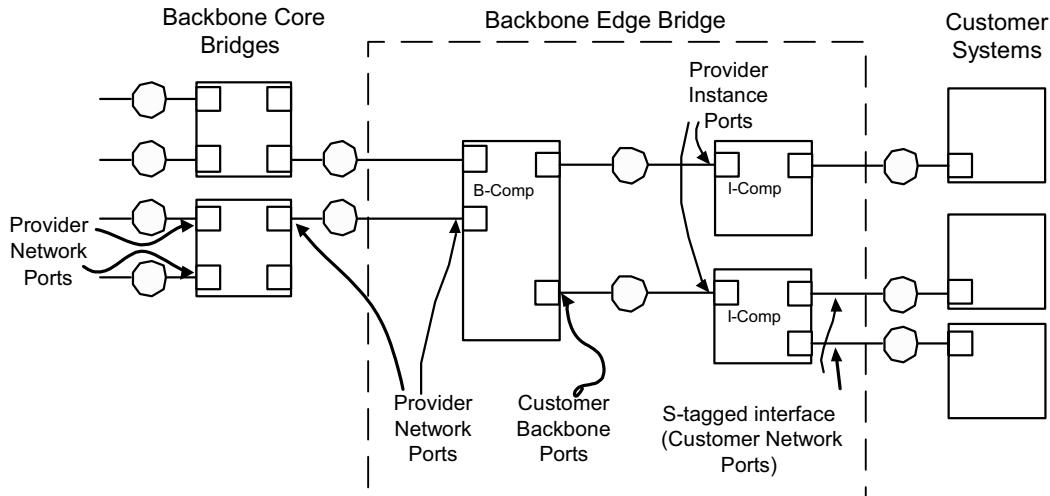


Figure 25-8—S-tagged service interface equipment

NOTE 2—It is always possible to build equipment that includes both Backbone Edge Bridge and Provider Edge Bridge components. In particular, it is possible for a BEB to support a C-tagged service interface for attachment to a C-VLAN bridged network. The resulting components are B-component to I-component to C-VLAN component.

25.5 I-tagged service interface

A PBBN may provide a native I-tagged service interface for attachment to another PBBN or for attachment to a customer's Provider Instance Port (PIP). An I-tagged service interface can provide access to all the backbone service instances within a PBBN. Access to backbone service instances is controlled by the configuration of the Customer Backbone Port (CBP) service instance tables (6.11). Each I-SID delivered over the I-tagged service interface by a customer identifies a service instance that will be carried over the PBBN. Service instances are carried over the PBBN inside a B-VLAN selected by the CBP. The customer must provide the B-DA address for frames delivered to an I-tagged service interface.

Figure 25-9 illustrates a customer network attached to a PBBN using an I-tagged service interface. The customer network uses a BEB with an I-component (26.6.1) or a B-component (26.6.2) connecting to the PBBN. The PBBN in turn is composed of BEBs interfacing to the customer and Backbone Core Bridges in the core of the PBBN used to forward frames between the BEBs.

An I-tagged service interface may be provided at a CBP of a BEB as illustrated by Figure 25-9 and Figure 25-10. In this service interface, the I-SID provided over the service interface within the I-TAG is mapped 1-1 to an I-SID within the PBBN. As illustrated, a customer (or peer) may attach to the I-tagged service interface through a CBP. In this configuration, each PBBN is attached at a CBP.

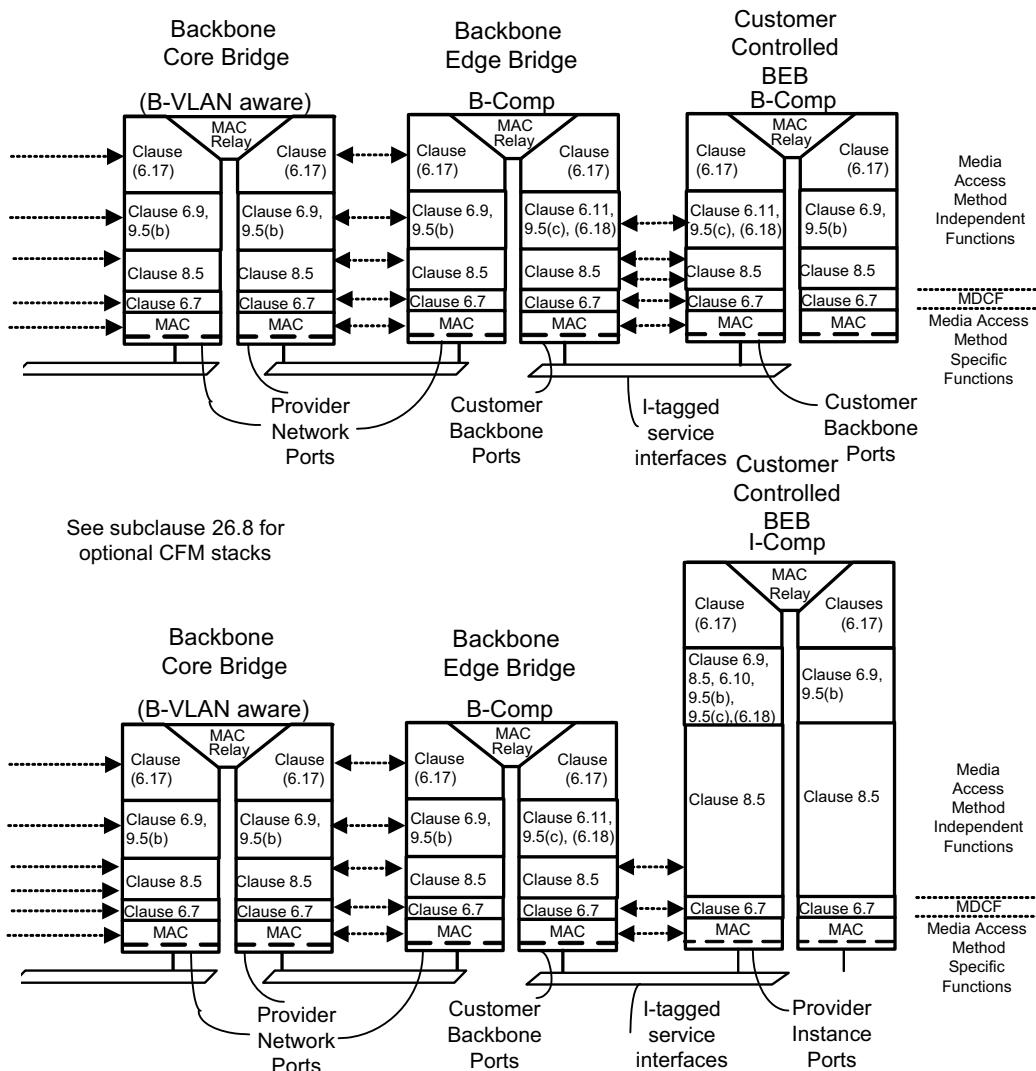


Figure 25-9—I-tagged service interface

Figure 25-10 shows an example of equipment used to implement an unprotected I-tagged service interface. For details on redundant connections and equipment, see 25.9. A B-component may support one or more I-tagged service interfaces. Each CBP may be associated with one I-tagged service interface. These CBP are connected to a customer system (or network) using an Ethernet LAN.

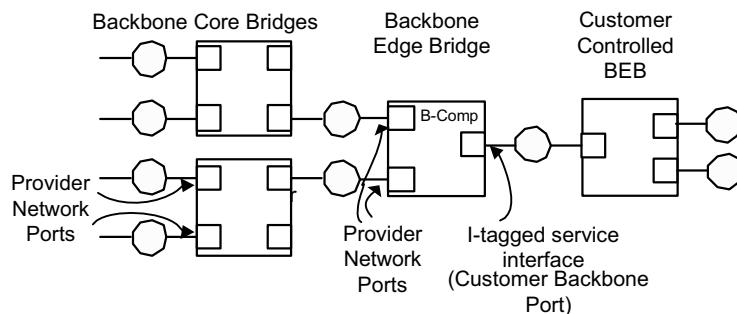


Figure 25-10—I-tagged service interface equipment

NOTE—A protected I-tagged service interface will have more than one PIP and can have a single backbone service instance, identified by an I-SID, enabled on more than one of those ports. This creates the potential for transit traffic on that backbone service instance to pass through the I-BEB without being delivered to a CNP. If this behavior is not desired, then the protected I-tagged service interface should be configured such that any single service instance is only enabled on a single PIP at any given time, or that only a single PIP is forwarding at any given time.

25.6 Service instance segregation

Segregation of data frames associated with different instances of MAC service is achieved by supporting each service instance with a backbone service instance identified by a backbone service instance Identifier (I-SID) and ensuring that

- a) No service frames are transmitted through a CBP without an I-TAG.
- b) No frames are accepted, i.e., received and relayed, from any customer system without first being subject to service instance selection.
- c) No frames are delivered to any customer system without explicit service instance identification.
- d) Prior to transmission through a PNP of a BEB, service frames are received either through an I-component's CNP or through a CBP. When the customer is attached to a CNP, the port is under the control of the backbone provider and is exclusively accessed by a single customer. In this case, the I-component used to support the CNP is under the control of the Backbone provider and is attached to a CBP through a PIP that is also under the control of the backbone provider and that may be shared with multiple customers. When the customer is attached to a CBP, the port is under the control of the provider and is exclusively accessed by a single customer. All frames received through the CNP or CBP (depending on how the customer is attached) must correspond to a service instance or instances that the customer is permitted to access. (The method used to assure the port is attached to the specified customer, and is therefore secure, is beyond the scope of this standard. See IEEE Std 802.1X.)
- e) The Backbone Core Bridges and the B-component of each BEB within the PBBN can only be controlled by the provider.

A single backbone service instance may support many customer service instances. Each customer service instance is supported by a single backbone service instance.

25.7 Service instance selection and identification

Service instance selection is provided to the attached customer system by the Port-based, S-tagged, or I-tagged service interfaces (25.3, 25.4, 25.5). In the Port-based interface, only one backbone service instance is offered to the attached customer and the CNP of the I-component is configured to accept only untagged or priority tagged frames. In all S-tagged service interfaces, the CNP of the I-component is configured with Enable Ingress Filtering (8.6.2) and the Port is only included in the Member Set for S-VLANs corresponding to service instances that the customer is permitted to use. The service instance for each frame received by the attached customer system is identified in the same way as frames transmitted using the same interface, but not necessarily in the same way that the service instance is selected or identified at other interfaces. In the I-tagged service interface, the CBP of the B-component is configured (12.16.5, 6.11) with the acceptable backbone service instances and I-SID values for use by the customer. A single backbone service instance can support any combination of Port-based, S-tagged, or I-tagged service interfaces.

NOTE—The means used by a backbone provider and a customer to determine the S-VIDs and I-SIDs used by the customer to select and identify a given service instance are outside the scope of this standard.

25.8 Service priority and drop eligibility selection

For all service interface types, the service priority is selected using the received priority for each frame. For S-tagged and Port-based service interfaces, the priority and drop eligibility may be regenerated using the Priority Regeneration Table (6.9.4). For I-tagged service interfaces, the priority and drop eligibility may be regenerated using procedures of 6.11.

The mechanism for determining the received priority varies with the type of service interface. Service priority selection is provided for Port-based service interfaces using the received priority signaled from the media access method of the port. If the media access method used to attach to the interface does not directly support priority, this will result in the selection of a single value for all frames. An attached system may also signal priority to a Port-based service interface on a per-frame basis by priority-tagging frames with an S-TAG that has a null VID (6.13).

Service priority selection is provided by S-tagged service interfaces using the PCP and DEI parameters conveyed in the S-TAG of each frame. An S-tagged service interface can provide a single backbone service instance for all S-VIDs received, and in this way functions like a Port-based service interface with the addition of the capability of delivering S-tagged service frames (whose S-TAG has a non-null VID).

Service priority selection is provided by I-tagged interfaces using the received priority decoded from the I-PCP and I-DEI fields in the I-TAG.

25.9 Service access protection

A customer system (or systems) that is part of a single PBN or that contains a customer operated I-component or B-component can attach to a PBBN using a protected service interface providing multiple LANs and (or) nodes, thus providing fault tolerance through redundancy of the service interface components. Three classes of service access protection may be supported (see Figure 25-11 and Figure 25-12), which provide for unprotected service interface, for link level redundancy at the access link or link and node redundancy at the access point. An access service interface may be a Class I, II, III access service interface.

Protected Port-based, S-tagged, and I-tagged service interfaces are depicted in Figure 25-11 and Figure 25-12, respectively.

Class I service interfaces depicted in these figures are unprotected service interfaces providing a single LAN and node. Class I service interface equipment is depicted in Figure 25-5, Figure 25-8, and Figure 25-10. These provide no redundancy and therefore will fail if any LAN or component fails.

Class II service interfaces are link protected service interfaces that provide multiple LANs; however, they do not provide any node redundancy. Class II service interfaces are tolerant of single or multiple LAN failures. These provide no node redundancy and therefore will fail if any node fails.

Class III service interfaces are link and node protected service interfaces providing multiple LANs and multiple nodes. Class III service interfaces are tolerant of single or multiple LAN failures and/or single or multiple node failures. The operation of a Class III service interface switches between access nodes within both the customer and provider networks; therefore, protection switching over a Class III service interface always results in state changes spreading over the customer and provider networks.

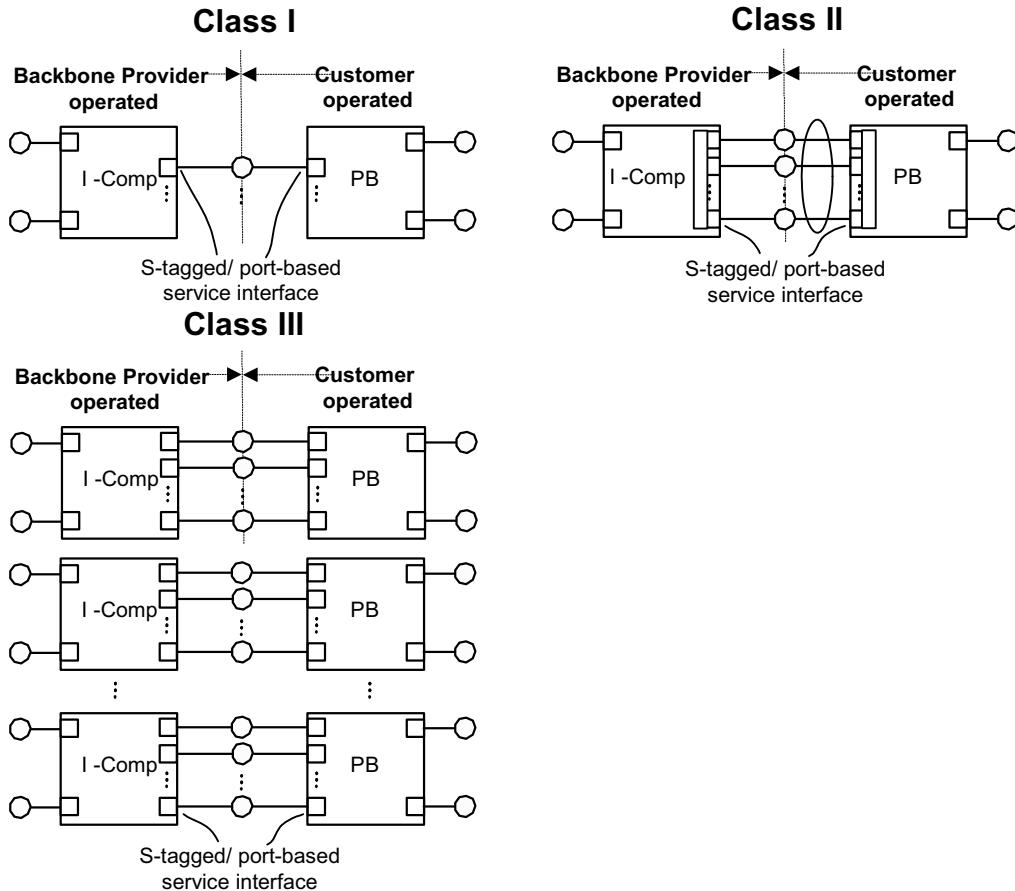


Figure 25-11—S-tagged and Port-based service interface access classifications

To attach a PBN to a PBBN the following rules must be maintained to assure loop-free operation over the extended network:

- Each PBN and PBBN prevents forwarding loops by running an independent spanning tree.
- Each PBN connects to other PBNs only through a PBBN.
- Each PBN ensures that no data frame passes through more than one BEB attachment point into or out of the PBBN. An attachment point is a single LAN connection to a PBBN. This connection may be part of a protected or unprotected Port-based (see Figure 25-4), S-tagged (see Figure 25-7), or I-tagged service interface (see Figure 25-9). A PBN may use L2GP (13.38) to assure only a single link is active when using a backup link to the PBBN.
- Each PBN ensures that it attaches any given S-VLAN to no more than one PBBN.

NOTE—When an attached PBN uses the L2GP (13.38) to enforce criterion c), then the PBBN may implement a Class I, II, or III protection interface. If the PBBN is configured for Class I or II protection, then it will provide multipoint services with the services delivered to two or more Class I or II interfaces for attachment to the PBN. In this case, the PBN is responsible for selecting one of the two attachments using the L2GP to assure criterion c). If the PBBN is configured for Class III protection, then the PBN must support MVRP over the access links to inform the PBBN which S-VIDs are active on which LAN connections.

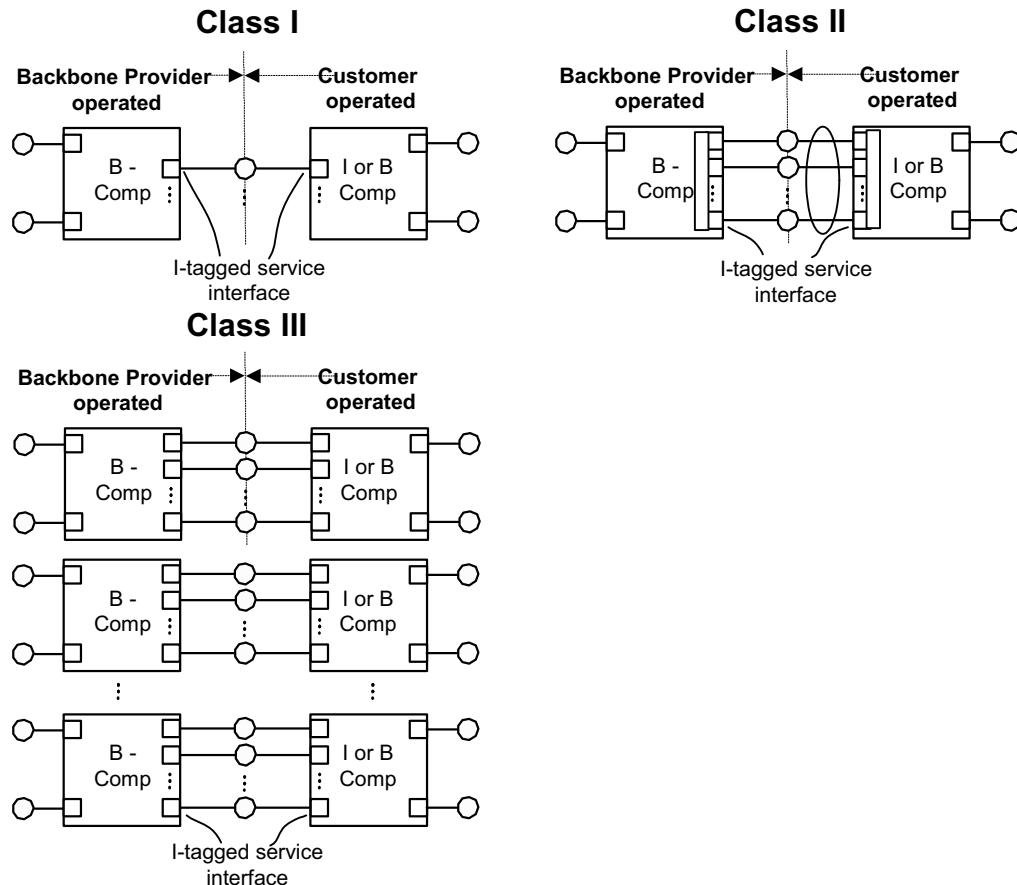


Figure 25-12—I-tagged service interface access protection classifications

25.9.1 Class II redundant LANs access protection

A Class II redundant interface is implemented using IEEE Std 802.1AX (Link aggregation). Using link aggregation, a single CNP and associated PNP of a Provider Bridge, or CBP and associated PIP, may be implemented using multiple LANs. A failure on any of the LANs implementing the link will cause all traffic to migrate to the remaining LAN. CFM may run both on the aggregate and on each individual link as shown in Figure 22-8.

NOTE—IEEE P802.1AX³⁷ is in progress and will replace IEEE Std 802.3-2005 Clause 43.

It is desirable for the distribution algorithm used on a Class II interface to divide traffic between the multiple links based on the service instance. One distribution algorithm that always retains a service on a single LAN is to place all the traffic on one of N links. In the event of a failure, all traffic is shifted to the highest priority remaining link. This distribution algorithm has the advantage that it provides for full reserved bandwidth on the backup link.

Another distribution algorithm that divides traffic by service and also allows traffic on all redundant links is to divide the traffic by either the S-VID or the I-SID. Distributing traffic based on the S-VID would be used for traffic over multiple LANs comprising an S-tagged interface, while distributing traffic based on the

³⁷This IEEE standards project was not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining a draft, contact the IEEE.

I-SID would be used for traffic over multiple LANs comprising an I-tagged interface. With either of these distribution schemes, services are allocated to one of the LANs used to implement the provider CNP or CBP and the customer PNP, PIP, or CBP. In the event of a failure, the services are moved to the remaining LANs.

25.9.2 Class III simple redundant LANs and nodes access protection

A Class III interface uses redundant LANs to connect a primary and one or more secondary customer nodes to a primary and one or more secondary BEBs. The operation protects the interface against failures on any of the interconnecting LANs or nodes used on each side of the interface. It is possible to use Class II protected interfaces for each link of the Class III interface. In this case, the Class II protection procedures must happen first and complete before a Class III protection switch begins.

For a Class III interface, each customer node is connected to a BEB using a single LAN. The interface is provisioned with a priority for each LAN which is part of the interface. Each of the LANs is constantly monitored using either PHY management, CFM, or both. In operation, a Class III interface uses only the highest priority operating LAN for traffic. The other LANs are protection paths. In a Class III interface, any switch in the LAN used for traffic will also switch both the customer node and the PBBN BEB since the connections of a Class III interface are sets composed of a customer node, a LAN, and a BEB.

When providing a Class III protected S-tagged interface, multiple CNPs, each on a different BEB, are used to create a single S-tagged interface. Each LAN of the S-tagged interface executes CFM at the port level to detect failures over the interface LANs. Information from CFM is used by the customer to determine which LANs of the S-tagged interface are operating and which are inactive. Each customer network runs MSTP to determine which S-VLANs are active on which LANs of the Class III interface. To avoid loops through the Class III interface, either the I-components participate in the customer network spanning tree (13.40), or the ports on the customer equipment that connects to the I-components are configured as Layer 2 Gateway Ports (13.38). Upon an active link failure between customer node and its corresponding BEB or upon the customer node failure, the customer network switches to one (or more) of the backup links and informs the PBBN which S-VLANs are active on which links using MVRP (11.2) on the access LANs. The BEB(s), upon receiving the MVRP message(s) from the customer network, generate the corresponding MRP-based message(s) for the affected I-SIDs and send these messages over the associated B-VLANs. The far-end BEBs, upon receiving these messages, will flush their C-MAC entries in their filtering database corresponding to the affected I-SIDs.

When providing a Class III protected I-tagged interface, multiple Customer Backbone Ports, each on a different B-BEB, are used to create the single I-tagged interface. In this scenario, each B-BEB is connected via a single link to the customer I-BEB where the link can be Class II protected. The customer I-BEB is in turn connected via multiple CNPs to the customer network. Each customer network runs MSTP to determine which S-VLANs are active on which LANs of the Class III interface. To avoid loops through the Class III interface, either the I-components participate in the customer network spanning tree (13.40), or the ports on the customer equipment that connects to the I-components are configured as Layer 2 Gateway Ports (13.38). PHY management or port level CFM are used to detect failures over the interface LANs between the customer I-BEBs and the customer network. When a failure is detected, the customer network informs the customer I-BEBs which S-VLANs are active on which links using MVRP (11.2). The customer I-BEBs in turn notify the provider B-BEBs using MRP-based messages corresponding to the effected I-SIDs. The provider B-BEBs relay this message to the other BEBs in the PBBN over the associated B-VLANs. The far-end BEBs, upon receiving these messages, will flush their C-MAC entries in their filtering database corresponding to the effected I-SIDs. Each LAN of the I-tagged interface also executes PHY management or CFM at the port level to detect failures over the I-tagged interface. In case of a complete I-tagged interface failure between the customer I-BEB and the provider B-BEB, the I-BEB uses this info to deactivate all its CNP ports toward the customer network.

25.10 Support of the MAC Service by a PBB-TE Region

A PBB-TE Region comprises of a continuous set of IB-BEBs and BCBs, capable of providing TESIs, that have allocated a common subset of B-VIDs to the control of an external agent (8.9) responsible for setting up Ethernet Switched Paths (ESPs). Each IB-BEB that belongs to a PBB-TE Region can provide interfaces that encapsulate customer frames in ESPs, thus allowing the PBBN operator to offer TESIs.

A PBB-TE Region providing the MAC service to attached stations is modeled as a sequence of relay functions, as illustrated in Figure 25-13. The MST Configuration Table associated with each BCB or B-component of an IB-BEB specifies a set of VIDs, each of which may be exclusively used throughout the PBB-TE Region to form part of an ESP identifier. Such a VID is called an ESP-VID and is associated with a special value of the MSTID in the MST Configuration Table, the TE-MSTID (8.9), indicating that the VID is not under the control of a spanning tree protocol.

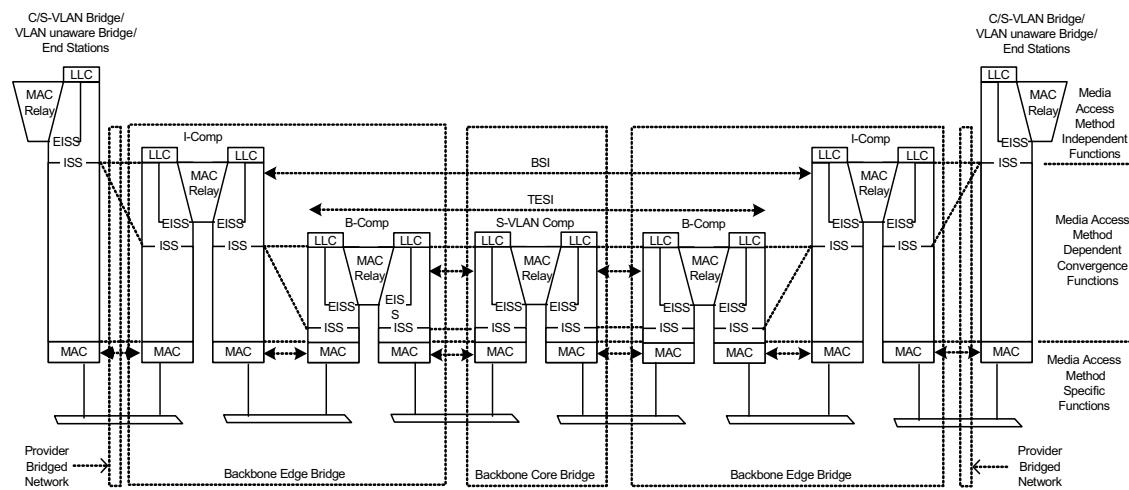


Figure 25-13—Internal organization of the MAC sublayer in a PBB-TE Region

The I-components of IB-BEBs supporting PBB-TE are responsible for encapsulating frames received from customers and allocating each frame to a backbone service instance and for decapsulating frames received from the network via backbone service instances. The allocation of customer frames to backbone service instances that will be transported by Traffic Engineered (TE) service instances is done at the PIP as described in 6.10. At the PIP the customer frame is encapsulated by an I-TAG, which includes the I-SID, and a set of source and destination backbone MAC addresses. The source backbone MAC address is the PIP MAC address that is configured to take the same value as the CBP MAC address of the internally connected CBP on the B-component. As a result, the backbone MAC addresses of frames associated with TESIs, the ESP MAC addresses, are always equal to the CBP MAC addresses.

CBPs on B-components in IB-BEBs supporting PBB-TE use the I-SID to map the encapsulated customer frames to TESIs or B-VLANs (6.11). Figure 25-13 depicts a TESI providing the MAC service at two CBPs. For an I-SID mapped to a TESI, the Backbone Service Instance table contains the terminating CBP's MAC address, the ESP-DA, and the ESP-VID of the associated ESP. I-SIDs mapped to a B-VLAN must not use a B-VID from the ESP-VID space. The B-Component MAC relay forwards frames belonging to an ESP between the CBP and PNPs or between PNPs using the ESP-DA and ESP-VID as described in 8.6.

25.10.1 Provisioning TESIs

ESPs are provisioned in a PBB-TE Region by an external agent. This replaces the spanning tree protocols and populates the filtering tables of the corresponding B-components and BCBs by creating static filtering table entries for the ESP MAC addresses and ESP-VIDs (Figure 25-14).

The ability of a PBB-TE Region to utilize an external management or control plane is facilitated by PBB because the PIP MACs and correspondingly the ESP MAC addresses can be managed by the Provider and therefore can all be identified in the Provider's topology by the external management or control plane.

The external PBB-TE management/control plane is responsible for maintaining and controlling all the active topology information to support point-to-point or point-to-multipoint Ethernet Switched Paths (ESP) over the PBB-TE Region. The PBB-TE active topology can co-exist with active topologies associated with spanning tree protocols and the allocation to a specific active topology is governed by the B-VID value. The PBB-TE external agent takes control of the ESP-VID range on the Bridges in a PBB-TE Region.

Similar to the forwarding connectivity created by the operation of spanning tree and learning, the PBB-TE management/control plane can form an active topology of CBP rooted trees, from each CBP belonging to a PBB-TE Region to a specific subset of any other CBPs in this region. These traffic engineered trees define the path(s) taken by the frames that belong to ESPs within the PBB-TE Region. Furthermore each such tree is further qualified by the PBB-TE reserved ESP-VID, thus enabling the construction of independent trees per ESP-VID. As a result each such CBP rooted tree can be identified by a CBP MAC address associated with the root of the tree, the CBP MACs associated with the leaves of the tree and the ESP-VID. The PBB-TE management/control plane routes the Ethernet Switched Paths along these trees by explicitly populating the Filtering Databases in the Bridges along each tree with the static filtering entries containing the CBP MAC DA and ESP-VIDs. The mechanism by which the PBB-TE management/control plane selects the path for a routing tree is not within the scope of this standard. The PBB-TE management/control plane also manages the bandwidth of all ESPs along each routing tree. For each destination CBP, which is part of a routing tree maintained by the PBB-TE management/control plane, PBB-TE will maintain tree(s) which provide symmetric paths from the CBP MAC DA to the CBP MAC SA. The ESP-VID(s) used in this reverse ESP(s) need not have the same ESP-VID used for the forward ESP. Accordingly each of the ESPs can be identified by a 3-tuple:

$\langle \text{ESP-DA}, \text{ESP-SA}, \text{ESP-VID} \rangle,$

where the three component fields are described as follows:

- a) The ESP-SA is the address of the Provider Instance Port (PIP) encapsulating the customer service instance that by configuration is the same as the address of the internally connected source CBP.
- b) The ESP-DA is identifying the CBP destination address(es); and
- c) The ESP-VID is a VID value distinguishing among ESPs having the same $\langle \text{destination}, \text{origin} \rangle$ pair. It can only take values that are allocated to the PBB-TE Region identified by the TE-MSTID (8.9).

An ESP is point-to-point or point-to-multipoint.

- d) A point-to-point ESP is an ESP where the ESP-DA and the ESP-SA in its 3-tuple identifier are individual MAC addresses.
- e) A point-to-multipoint ESP is an ESP between one root CBP to n leaf CBPs, identified by a 3-tuple where the ESP-DA is a group MAC address identifying the n leaves CBPs, and the ESP-SA is the individual MAC address of the root.

A TESI, is an instance of the MAC service supported by a set of ESPs, collectively identified by a single TE-SID, forming a bidirectional service. The following two types of TESIs are described:

- f) A point-to-point TESI that is supported by two point-to-point ESPs where the ESPs' endpoints have the same CBP MAC addresses.
- g) A point-to-multipoint TESI that is supported by a set of ESPs that comprises one point-to-multipoint ESP from a root to each of n leaves plus a point-to-point ESP from each of the n leaves to the root.

NOTE—For reasons relating to TESI monitoring diagnostics, operational simplicity, etc., this standard assumes the point-to-point ESPs associated with a point-to-point TESI are symmetric and that the point-to-point ESPs associated with a point-to-multipoint TESI are routed from each of the n leaves to the root along the branches of the point-to-multipoint ESP. Support for a TESI which comprises ESPs not routed in this way is problematic, and is not defined in this standard.

Although an ESP is identified by the 3-tuple <ESP-DA, ESP-SA, ESP-VID> it is only the ESP-DA, ESP-VID pair that are used for forwarding decisions. The combination (ESP-DA, ESP-VID) is treated as though it was a single 60 bit address, where 12 bits are the ESP-VID and 48 bits are the ESP-DA. This ESP-VID field is used as a path selector to the destination CBP rather than a B-VLAN ID, allowing up to 2^{12} unique routing trees to any single CBP. In Figure 25-14, two paths are configured to reach S2. These two paths are separated by using a different ESP-VID in combination with the ESP-DA for a second path.

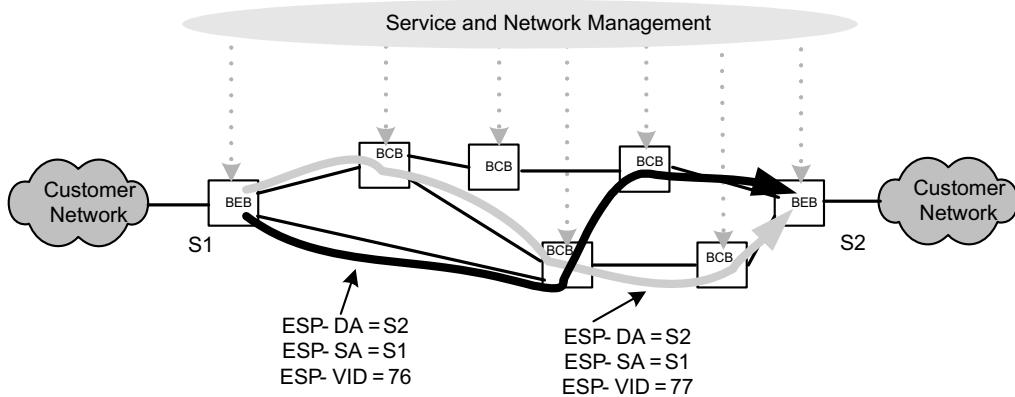


Figure 25-14—PBB-TE Region

If the ESP-VID range delegated is the 4094 possible values, then each CBP termination can sink ($2^{12}-2$) routing trees, and the theoretical network maximum is about 2^{60} ESPs. The number of ESPs supported by PBB-TE is constrained by issues related to coordinated management, independent management of groups of nodes or independent routing choices through parts of the path, limit through intermediate nodes etc. In the cases where B-VLAN services are also offered by the same network, it is expected that most B-VIDs will be allocated to those B-VLANs since the use of the MAC addresses as part of the ESP identifiers allows at least 2^{48} ESPs and the ESP-VIDs can be used mainly to provide path diversity.

25.10.2 ESP forwarding behavior

Forwarding of frames belonging to an ESP-VID differs from forwarding of frames belonging to B-VLANs in the following ways:

- a) Frames associated with the ESP-VID are discarded when a static filtering entry corresponding to the (ESP-DA, ESP-VID) is not found in the FDB [item b) in 5.4.3]. Proper operation of PBB-TE requires the provisioning of an FDB entry in each Bridge through which an ESP passes in order to specify the forwarding of traffic associated with that ESP. Such an entry may be absent due to a provisioning error. In the event of such an error, it is important that frames associated with the improperly configured ESP not be flooded, as PBB-TE traffic engineering would be violated if traffic intended for a particular ESP is sent on one or more other ESPs. Further, flooding based on

the ESP-VID may result in unbounded looping and replication as the ESP-VID is not associated with a spanning tree. For this reason, frames associated with the ESP-VID are discarded when a static filtering entry corresponding to the (ESP-DA, ESP-VID) is not found in the FDB. Such an ESP MAC address may represent an Individual Address, for which no more specific Static Filtering Entry exists or a Group Address, for which no more specific Static Filtering Entry exists (8.8.1).

- b) MAC learning is not performed on receipt of a frame carrying an ESP-VID [item a) in 5.4.3]. Where ESPs have been correctly provisioned, learning would not interfere with the proper operation of Bridges, but in the presence of provisioning errors, learning may result in undesired behavior.
- c) The active topology for the ESP-VIDs allocated to PBB-TE is no longer controlled by spanning tree protocols but by an external agent that is responsible for setting up the ESPs [item b) in 5.4.3].

NOTE 1—In a network supporting both VLAN (i.e., PBB) and ESP (PBB-TE) traffic, the establishment of an ESP and the policing of traffic at the ingress of the ESP are necessary but not sufficient conditions to ensure that traffic associated with an ESP receives the QoS intended by the network operator. The operator should ensure that the QoS requirements of an ESP are satisfied and that sufficient network capacity is available to accommodate broadcast-unknown traffic, spanning tree control traffic, and other traffic associated with VLAN usage.

Key properties of an ESP are as follows:

- d) The ESP is identified at all points along its path by a single identifier <ESP-DA, ESP-SA, ESP-VID>.
- e) The information referenced for forwarding <ESP-DA, ESP-VID> is contained within the ESP Identifier.
- f) The information referenced for forwarding <ESP-DA, ESP-VID> does not change along the length of the ESP.

NOTE 2—The static and provisioned nature of an ESP provides support for traffic engineering. These properties of the ESP, together with policing and queueing functions supported by a Bridge (8.6) can provide per ESP QoS.

25.11 Transparent service interface

A Provider Backbone Bridged Network PBBN may provide a transparent service interface for customer attachment. A transparent service interface is delivered on a Customer Network Port provided by a Backbone Edge Bridge (BEB) as illustrated in Figure 25-15 and Figure 25-16. A transparent service interface may attach to Provider Bridges, C-VLAN Bridges (5.9), IEEE 802.1D bridge, router, or end-station. The service provided by this interface forwards all frames over the backbone on a single backbone service instance.

A transparent service interface does not recognize, add, remove, or modify either S-tags or C-tags on customer frames. It does not learn customer addresses, nor does it filter frames based on any MAC addresses except the TPMR component Reserved addresses listed in Table 8-3 and the management address of the TPMR component.

Figure 25-15 illustrates a customer network attached to a PBBN using a transparent service interface. The customer network uses End Stations, Customer Bridges, or Provider Bridges for connecting to the PBBN.

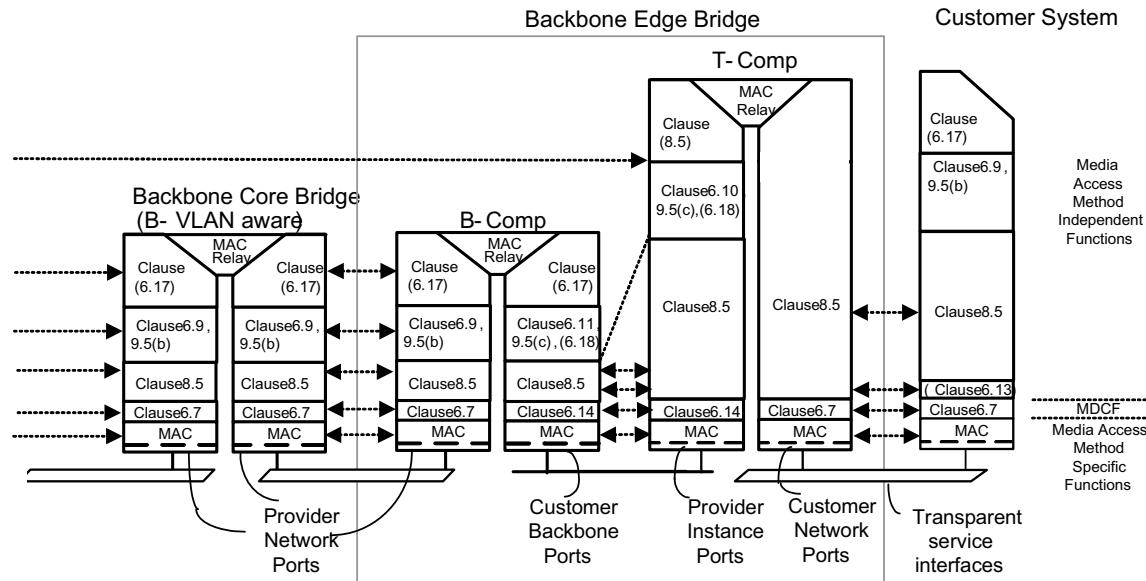


Figure 25-15—Transparent service interface

Figure 25-16 shows an example of equipment used to implement a transparent service interface. In this diagram a BEB is formed by connecting two T-components and a B-component. These connections may be over a backplane or over a LAN. A T-component supports a single transparent service interface associated with a CNP. A CNP is connected to a customer system (or network) using a LAN.

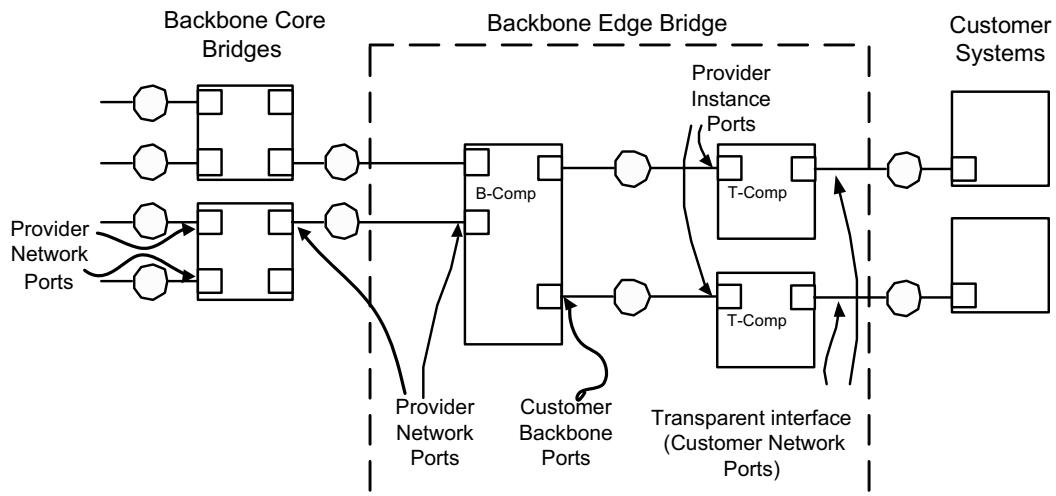


Figure 25-16—Transparent service interface equipment

26. Principles of Provider Backbone Bridged Network operation

This clause establishes the principles and a model of Provider Backbone Bridged Network (PBBN) operation. It provides the context necessary to understand how the

- a) Operation of individual Backbone Edge Bridges (BEBs) and Backbone Core Bridges (BCBs) (Clause 8, Clause 25);
- b) Configuration and management of individual BEBs and BCBs (Clause 12);
- c) Management of Spanning Tree and VLAN topologies within a PBBN (Clause 7, Clause 11, and Clause 13)

support, preserve, and maintain the quality of each of the instances of the MAC Service offered to the customers of the PBBN (Clause 6, Clause 25) including the following:

- d) Independence of each service instance supported by a PBBN from other service instances (25.6).
- e) Identification of service instances and backbone service instance within the PBBN (25.7, 8.8).
- f) Maintenance of service availability in the event of the failure, restoration, removal, or insertion of LAN components connecting a customer network to a PBBN (Clause 6, Clause 11, Clause 13, 25.9).

A PBBN is a Virtual Bridged Local Area Network that comprises BEBs at the edge of the PBBN and BCBs at the core of the PBBN and attached LANs, under the administrative control of a single backbone provider. The principal elements of PBBN operation are those specified in Clause 16, as amended by this clause.

NOTE 1—The term PBBN is used exclusively to refer to networks configured and managed as specified by this clause and comprising only (a) BCBs and BEBs and (b) communications media and equipment providing the Internal Sublayer Service (6.6). While the requirements of Clause 25 are generally applicable to similar services, a generalized framework for all the network designs that could support these requirements, while useful in the context of other equipment and services, is outside the scope of this standard. This clause describes specific best practice for PBBNs to ensure that the requirements for bridge functionality are clear. Conformance of a BEB implementation to this standard does not require that the implementation be used as specified in this clause; merely, that it is capable of being so used.

NOTE 2—Within a PBBN, an instance or instances of the MAC Service are reserved for the backbone provider's own use to configure and manage the network. All frames associated with such service instances, and that are not confined to an individual LAN, are subject to service instance selection, segregation, and identification as specified in 26.1.

26.1 Provider Backbone Bridged Network overview

The principal elements of Provider Backbone Bridged Network (PBBN) operation comprise the following:

- a) Encapsulation/de-encapsulation of service frames in Backbone MAC Frames (6.10).
- b) Service instance segregation within a PBBN for service frames (25.6).
- c) Service instance selection on ingress, and service instance identification on egress, for each customer frame (25.7).
- d) Service access protection (25.9).
- e) Resource allocation and configuration to provide service instance connectivity (26.3).

The following may also be included:

- f) Management of customer end station address learning (26.4).
- g) Prevention of connectivity loops formed through attached networks (26.5).
- h) Scaling of PBBNs (26.6).
- i) Provisioning of PBBN services (26.7).
- j) Management of PBBN service connectivity (26.8).

26.2 Provider Backbone Bridged Network example

An example Provider Backbone Bridged Network (PBBN) is illustrated by the Figure 26-1. The example network is divided into three major areas. These are the Customer Equipment area, which is owned by the customer and is interfaced to the PBBN; the PBBN area, which is owned by the backbone provider and interconnects to the customer equipment and to other PBBNs; and the other PBBN area, which is owned by the same or other backbone providers and interconnects to multiple Provider Bridged Networks allowing the network to scale indefinitely.

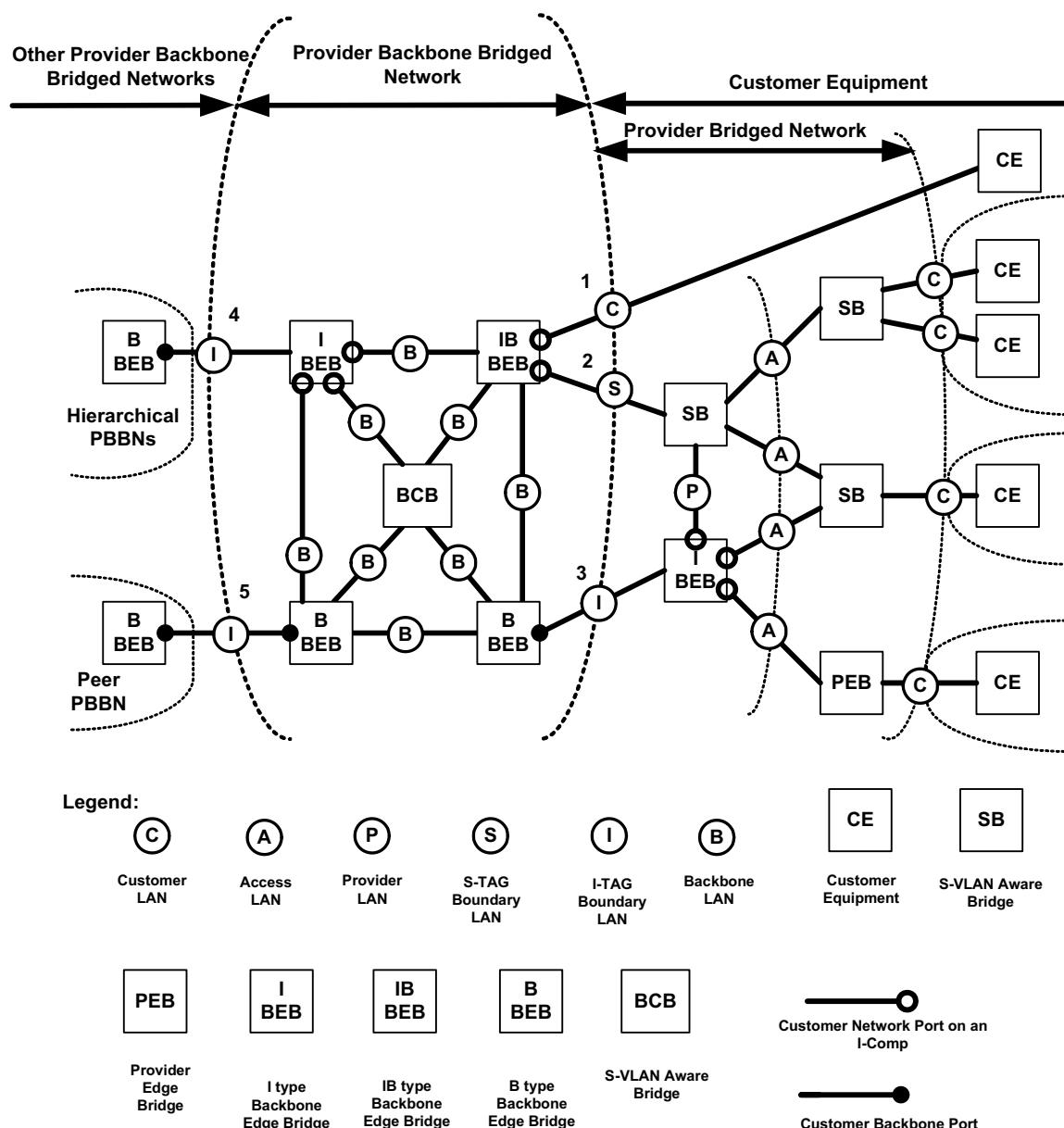


Figure 26-1—PBBN example

The interface between a BEB and a Provider Bridge is a Customer Network Port (CNP) of an I-Component as specified in 6.10. The interfaces between BEBs and Backbone Core Bridges and between Backbone Core Bridges and Backbone Core Bridges are Provider Network Ports (PNPs). In Figure 26-1, these interfaces are over LANs labeled S and B. The interfaces between B-BEB and I-BEBs, or between B-BEBs in different PBBNs, are over LANs labeled I. All exterior interfaces from a PBBN are over S, I, or C LANs; whereas, all interior interfaces within a PBBN are over B LANs. The S, I, and C LANs (exterior interfaces) are at the boundary between the PBBN and the attached customer networks.

The B LANs (interior interface) are backbone LANs, which interconnect all the Provider Backbone Bridges within a PBBN.

Within the network example BEBs, Backbone Core Bridges and B LANs are secured so that only the backbone provider operating the PBBN can manage the reception, transmission, and relay of frames between BEBs and BCBs. The arbitrary physical network topology of the PBBN and the connectivity that it provides to support segregated instances of the MAC Service is designed and managed by the backbone provider to meet bandwidth and service availability requirements at the Customer Backbone Ports (CBPs) and PIPs. Application of the service instance ingress and egress rules at these Ports in support of service instance selection and identification ensures that frames cannot be transmitted or received on any backbone service instance by any customer's equipment without prior agreement with the provider.

The active MSTP topology of the PBBN area is separated from the active topology of the customer equipment area. This is accomplished by isolating the MST BPDUs for each customer network from the PBBN at the BEBs that surround the perimeter of the PBBN. The BEBs also stop the propagation of MST BPDUs, used to support the active topology of the PBBN, into the Provider Bridged area. In Figure 26-1, Provider Bridge MST BPDUs are propagated over the A, P, C, and S LANs but not over the I or B LANs, while Provider Backbone Bridge MST BPDUs are propagated over the B LANs but not over the I, C, A, P, or S LANs.

For Port-based service interfaces, the entire C LAN is treated as a single instance of the MAC service. The BEBs extend the C LAN over the backbone by mapping all service frames on a single backbone service instance identified by a Backbone Service Instance Identifier (I-SID). The I-SID used is provisioned by the provider operating the PBBN. The provider sets the I-SID on each BEB port attached to a C LAN that is part of the port based service. The BEBs encapsulate the service frames with an I-TAG, B-TAG, B-SA, and B-DA. During the encapsulation, any C-TAG delivered to the CNP is retained in the encapsulated frame as part of the Service Data Unit. The BEBs then map the frames onto a B-VLAN with its B-VID contained in the B-TAG and which interconnects the BEBs provisioned for the service. These new frames are transmitted over the B LANs by the BEBs and by the BCBs that carry the B-VLAN. Since the initial octets of the data conveyed in each backbone frame comprise a Service VLAN tag, the frames may be forwarded by BCBs of the PBBN until they reach the next BEB where they might be de-encapsulated. During de-encapsulation, the B-DA, B-SA, B-TAG, and I-TAG are stripped. The C-DA and C-SA from the I-TAG will become the DA and SA transmitted to the receiving C LAN. If the encapsulated SDU contains a C-TAG, it becomes the outside tag as the frame is transmitted to the receiving C LAN.

For S-tagged service interfaces, each instance of MAC service is carried over the S LANs on one or more S-VLANs. The BEBs preserve the S-VLAN over the backbone by mapping them onto a Backbone Service Instance Identifier (I-SID) and in the case of S-VLAN bundling carrying the S-TAG. This operation is performed by the backbone provider operating the PBBN by configuring the VIP-ISID on each BEB attached to an S LAN. The BEB maps the S-VID to I-SID and encapsulates the original service frame with a new I-TAG, B-SA, and B-DA. The BEB then maps the frame onto a B-VLAN, which interconnects BEBs. This new frame is transmitted over the B LANs by the BEBs and by the Backbone Core Bridges. Since the initial octets of the data conveyed in each backbone frame comprise a Service VLAN tag, the frames may be forwarded by Backbone Core Bridges of the PBBN until they reach the next BEB where they might be de-encapsulated.

For I-tagged service interfaces, each instance of MAC service is identified by an I-SID value carried over the I LAN in an I-TAG. The BEB maps the frame onto a B-VLAN, which interconnects BEBs. This new frame is transmitted over the B LANs by the BEBs and by the Backbone Core Bridges. Since the initial octets of the data conveyed in each backbone frame comprise a Service VLAN tag, the frames may be forwarded by Backbone Core Bridges of the PBBN until they reach the next BEB where they might be de-encapsulated.

Backbone MAC addresses are used to identify the destination BEB's PIP. These backbone MAC addresses are learned by each PBB bridge as frames are exchanged over B-VLANs. To perform the encapsulation and de-encapsulation of service frames, BEBs use the connection identifier stored in the filtering database (see 8.8) to correlate Customer MAC Addresses to backbone MAC addresses. At startup, the BEBs have not learned the B-MACs or the C-MAC addresses yet. When the B-MAC address is unknown, the BEB encapsulates the service frames using the Default Backbone Destination address. The same holds true for the service frames having a group or broadcast destination address. An individual B-DA is used for forwarding unknown or group frames when the service is point-to-point and where the far end individual address is known. Both the B-MAC and C-MAC addresses are learned by the provider BEB (see Clause 8).

Frames containing encapsulated service data are carried within the B-VLANs created over the PBBN. It is also possible to carry unencapsulated S-VLAN traffic in the PBBN core by allocating some of the PBBN VID space to S-VLANs. The B-VLAN determines the route that the frames will take and limits broadcasting within the backbone. The B-TAG is added to the frame at the CBP. The selection of B-VLAN used to form the B-TAG is determined by the configuration of the CBP Backbone Service Instance table. This table maps I-SIDs to B-VIDs and is created as part of service provisioning.

If a BEB's PIP receives a Backbone MAC frame whose B-SA is the same as the Backbone MAC address of the PIP, then a loop exists through the BEBs. Optionally, the BEB may use this condition for loop detection.

26.3 Backbone VLAN connectivity

The B-VLAN Topology of each B-VLAN is established by the mechanisms introduced in 7.1 and Figure 7-1. The backbone provider can use and configure MSTP to provide a number of independent spanning tree active topologies and can assign each B-VLAN to one of these active topologies to best use the resources in the network. MVRP running in the context of each spanning tree active topology configures the extent of each B-VLAN to the subset of that active topology necessary to support connectivity between the customer points of attachment to the instance of MAC service provided, and can reconfigure that connectivity as required if the spanning tree active topology changes.

The operation of MSTP within a Backbone Provider's network is independent of the operation of any spanning tree protocol within attached provider or customer networks. This is achieved by removing all MST BPDUs received or to be transmitted at the service access interfaces.

The operation of MVRP within a Provider Backbone Bridged Network (PBBN) is independent of the operation of any configuration protocol within attached customer networks. The Provider Bridge MVRP address (Table 8-1) is used as the destination address of all MRPDUs transmitted in support of the MVRP Application. Frames received by CBPs and addressed to the Customer Bridge MVRP address (Table 10-1) are subject to service instance selection and relay in the same way as service frames. The MVRP Administrative Control for each B-VLAN is either Registration Fixed or Registration Forbidden on all CBPs, so no information is received from any Backbone Edge Bridge MVRPDU that has been erroneously transmitted by a customer system.

26.4 Backbone addressing

When customer frames enter a PBBN they are encapsulated by the addition of backbone MAC addresses and the creation of an I-TAG. The encapsulation is the result of the actions of a PIP as specified in 6.10. The PIPs represent the endpoints of a backbone service instance, and the backbone MAC addresses identify these endpoints. (More specifically the Virtual Instance Ports of the PIP represent the endpoints of a backbone service instance, and the combination of the backbone MAC addresses and the I-SID identify these endpoints. Since the I-SID identifies which VIP of a PIP is the backbone service instance endpoint, the backbone MAC address need only identify the PIP. Therefore, all VIPs of a PIP can share the same MAC address.) This encapsulation ensures that only I-components learn Customer MAC Addresses, while B-components and bridges in the PBBN core learn only backbone MAC addresses.

The backbone source address (B-SA) of an encapsulated frame identifies the PIP that performed the encapsulation and is referred to as the PIP MAC address. For a PIP that connects to an external LAN, this is the address of the PIP itself. For a PIP that has an internal connection to a CBP in the same Backbone Edge Bridge, this can be any address that results in delivering received I-tagged frames to the I-component, such as the Bridge address of that I-component. When the PIP MAC address is also used by CFM MPs instantiated on the PIP, there are additional constraints on the uniqueness of that address (26.4.3).

The backbone destination address (B-DA) of an encapsulated frame identifies the PIP(s) to which the frame should be delivered. The default value for the B-DA is the Backbone Service Instance Group address constructed by concatenating the three octet OUI shown in Table 26-1 with the three octet I-SID, and asserting the I/G bit in the first octet of the resultant value to signify a group MAC address. In a Backbone Service Instance Group address formed in this way, the OUI value is encoded in the first three octets of the address, with the most significant octet of the OUI encoded in the first octet, with the I/G bit (the LSB) set, and the least significant octet of the OUI encoded in the third octet; the most significant octet of the I-SID is encoded in the fourth octet of the address and the least significant octet of the I-SID in the sixth octet.

Table 26-1—Backbone Service Instance Group address OUI

Name	Value
IEEE 802.1Q Backbone Service Instance Group address OUI	00-1E-83

NOTE—Table 26-1 uses the Hexadecimal Representation specified in IEEE Std 802.

When the B-DA of a frame is the Backbone Service Instance Group address, the nominal behavior of the PBBN is to deliver the frame to all CBPs reachable within the B-VLAN to which the backbone service instance is mapped. Filtering based on I-SID by the egress CBP assures that frames are not transmitted by CBPs that are not part of the backbone service instance. Flooding frames throughout the B-VLAN and filtering at egress provides the correct connectivity for the backbone service instance, but may result in inefficient use of PBBN resources. More efficient frame delivery can be achieved by configuring Static Filtering Entries in the backbone bridges, or using MMRP to create MAC Address Registration Entries, that restrict the delivery to just the endpoints of the backbone service instance. More efficient delivery can also be achieved by learning individual backbone MAC addresses at a PIP, and in some cases by configuring the CBPs to translate the Backbone Service Instance Group address.

26.4.1 Learning individual backbone addresses at a PIP

Individual backbone addresses are learned by means of the connection_identifier parameter generated by the PIP and stored in the Filtering Database. For each received frame, the PIP includes in the EM_UNITDATA.indication a connection_identifier parameter that identifies the backbone source address of

the frame. When the I-component Learning Process creates or updates a Dynamic Filtering Entry for a customer source address (C-SA) of a frame received through a PIP, it stores the connection_identifier parameter of the EM_UNITDATA.indication for that frame. The connection_identifier parameter is included in the EM_UNITDATA.request for subsequent frames destined to that customer address. Within the PIP, the connection_identifier determines the individual backbone address associated with the customer address. This individual address is then used as the B-DA in place of the Backbone Service Instance Group address.

When a backbone service instance includes only two VIPs, it is possible to use a learned individual backbone address as the default B-DA instead of the Backbone Service Instance Group address. This is accomplished by configuring the AdminPointToPointMAC parameter for the VIP to ForceTrue, which results in the operPointToPointMAC parameter having a value of TRUE. In this case, the PIP copies the B-SA of frames received for that VIP (i.e., the individual MAC address of the peer PIP) to the Default Backbone Destination parameter of the VIP. Subsequent frames transmitted by the PIP on that backbone service instance will then use the individual MAC address of the peer PIP as the B-DA even if the C-DA is unknown or broadcast. Enabling the enhanced filtering criteria (8.7.2) on a VIP will prevent the I-component from creating Dynamic Filtering Entries for Customer MAC Addresses unnecessarily when the backbone service instance is point-to-point and the S-VLAN that maps to the backbone service instance has only the VIP and one CNP in its member set.

When the enableConnectionIdentifier of a VIP is set to FALSE, the connection_identifier is kept to a NULL value and correspondingly the B-DA used by frames associated with this VIP is always the Backbone Service Instance Group address.

26.4.2 Translating backbone destination addresses at a CBP

In some cases, when the B-DA is the Backbone Service Instance Group address it may be advantageous that the CBP translate this to a different address. Examples where such translation may be useful include the following:

- a) When a backbone service instance has only two endpoints within a PBBN, the Backbone Service Instance Group address may be translated at the ingress CBP to the individual address of the egress CBP.
- b) When multiple backbone service instances connect the same set of CBPs, the ingress CBP may translate the Backbone Service Instance Group address to a single group address chosen for the set of backbone service instances.

These translations can be accomplished by configuring the Default Backbone Destination parameter in the service instance table of the CBP (6.11). Whenever an ingress CBP is configured to translate the B-DA, the egress CBP must be configured to translate the B-DA back to the Backbone Service Instance Group address. This allows multiple PBBNs connected through Peer E-NNIs to use addresses local to each PBBN.

The CBP is also capable of translating the I-SID on frames entering and exiting the PBBN. This allows the values of the I-SID to be local to a PBBN. If the B-DA of the frame is the Backbone Service Instance Group address and the CBP translates the I-SID, the CBP will also reconstruct the Backbone Service Instance Group address using the new value of the I-SID.

26.4.3 Backbone addressing considerations for CFM Maintenance Points

Possible placement of CFM shims in a PIP and a CBP are shown in Figure 26-2. In the PIP, these include MPs to monitor backbone service instances (identified by I-SIDs) in the PBBN, as well as MPs to monitor service instances (identified by VIDs of S-VLANs) in the attached PBNS. In the CBP, these include MPs to monitor backbone service instance (identified by I-SIDs) in the PBBN, as well as MPs to monitor B-VLANs in the PBBN. All of these MPs are assigned individual addresses subject to the constraints specified in 19.4; however, there are additional considerations for the addresses used by MPs in PIPs and CBPs.

In a PIP, the individual address assigned to the backbone service instance level MPs must be unique within the PBBN. The individual address assigned to the S-VLAN level MPs must be unique within the set of PBNs interconnected by any backbone service instance supported by the PIP. Therefore these two groups of MPs may use a common address that is globally unique; however, they may need distinct addresses if local addressing is used in the PBBN.

In the CBP, both the backbone service instance level MPs and the B-VLAN level MPs require an individual address that is unique within the PBBN. Therefore these two groups may use a common address whether it is a global or local address.

26.5 Detection of connectivity loops through attached networks

The transmission and reception of MST BPDUs through CNPs will detect accidental direct connection of those ports. A backbone provider cannot rely on any customer network relaying such frames, and should develop a policy and mechanisms to deal with potential data loops that can arise if the attached customer systems do not operate their own instance or instances of a spanning tree protocol.

A bridged network can be redundantly attached to a PBBN by the means of several Layer Two Gateway Ports (13.38). By way of communication within their region and without any influence from the PBBN, those ports will provide connectivity to the PBBN through a single Layer Two Gateway Port, thus avoiding any bridging loop between the PBBN and the network.

NOTE 1—Use of the restrictedRole parameter at ingress ports ensures that receipt of BPDUs addressed to the Provider Bridge Group address cannot disrupt internal connectivity within the PBBN.

NOTE 2—Specification of PBBN policies, mechanisms, and heuristics used to detect or minimize the impact of data loops created by customer systems is not addressed by this standard. They can include, but are not limited to, bandwidth limitation, charging policies, detection of the repetitive movement of the apparent location of customer stations, and customer agreement to allow the use of PBBN loop detection protocols by not filtering the associated frames.

NOTE 3—A data loop is not the only possible cause of excess bandwidth consumption by a given customer of a PBBN, and the PBBN is usually required to meet service guarantees to other customers irrespective of the cause of the excess bandwidth demand. Data loops are not a unique threat to satisfactory overall network performance. Their distinct characteristic is consumption of discretionary bandwidth without benefiting any customer. The customer that creates the loop can suffer particularly serious network degradation or excess cost as the PBBN limits the total bandwidth consumed by that customer. It is therefore in the interests of each individual customer and the PBBN to raise service satisfaction by preventing and detecting loops.

26.6 Scaling of Provider Backbone Bridges

For PBBN deployments, it is important to be able to interconnect PBBNs operated by different organizations. The interfaces between PBBNs are examples of the Metro Ethernet Forum (MEF) External Network to Network Interface (E-NNI) referred to in the MEF reference model (MEF 4 Clause 7.2 and Figure 3). E-NNIs for interconnection of PBBNs can be further classified into two types: hierachal interconnect and peer interconnect. Examples of hierachal and peer interconnects are depicted in Figure 26-1 at the interfaces marked 4 and 5.

26.6.1 Hierachal PBBNs

The hierachal interconnect connects the PBBNs in a leveled hierarchy. In this interconnect model one PBBN acts as a second level ($n + 1$) PBBN to a first level (n) PBBN. This model allows scaling the number of services by additional hierachal level. Each level of a hierachal interconnect adds an additional encapsulation layer.

In the hierarchical model, the $n + 1$ level PBBN carries the n -level PBBN B-VLANs as services in the same manner that the first level PBBN carries the customer S-VLANs as services. In this model an S-VLAN is presented to a level n PBBN for transport. The PBBN transports the S-VLAN as an instance of MAC service within a B-VLAN generated by the n -level PBBN. The n level PBBN is connected to an $n + 1$ level PBBN over a hierachal E-NNI interface. The $n + 1$ level PBBN carries B-VLANs from the attached n -level PBBN as service instances in $n + 1$ level B-VLANs through the $n + 1$ level PBBN.

Since each hierachal level requires an additional encapsulation, the nesting depth is limited by the maximum defined frame length for the MAC type used.

26.6.2 Peer PBBNs

The peer interconnect between PBBNs results in a flat service space that does not increase the encapsulation nesting depth. The network scales the number of services over the combined PBBNs by service localization. Further localization and scaling of services is supported by the I-SID filtering functions in addition to the optional I-SID translation functions of the CBP interfaces on each side of the peer E-NNI (6.11). Services that are limited in extent to a single PBBN do not consume any resources within the other connected PBBNs.

The peer interconnect will extend backbone service instances across the peer connected PBBNs. In the peer interconnect a service instance in PBBN A is connected to a service instance in PBBN B by the peer E-NNI between the PBBNs. The peer E-NNI is formed by an I-tagged Service Interface between two CBPs (6.11), one in each of the peer PBBNs. The I-SID used over the E-NNI must be agreed to mutually by both providers. At each end of the E-NNI the CBP may optionally translate the I-SID (and the B-DA if it is the default multicast address generated from the I-SID) used over the E-NNI to a new I-SID for use within the PBBN. The CBPs also map the I-SID to a B-VLAN within the PBBN, as no B-TAGs are transmitted across the peer interconnect.

26.7 Network Management

Management of a Provider Backbone Bridge is directly under the control of the backbone provider. PBN customers shall not have access to managed objects related to elements of Provider Backbone Bridges within the PBBN.

In SNMP management systems, this can be realized by implementing one management entity acting as an agent in the Provider Bridge and separating the management views that include customer networks information from the management views that include provider networks information using the mechanisms defined by the SNMP management framework in IETF RFC 3411.

26.8 Connectivity Fault Management in Provider Backbone Bridges

Connectivity Fault Management (CFM 18, 19, 20, 21, 22) may operate over LANs, C-VLANs, S-VLANs, B-VLANs, and backbone service instances identified by I-SIDs. CFM Maintenance domain End Points (MEPs) and Maintenance domain Intermediate Points (MIPs) may be inserted at any of the EISS service interfaces described by 6.9, 6.10, and 6.11, at ISS service interface described in 6.6, and within the EISS and Backbone Service Instance Multiplex Entity described in 6.17 and 6.18.

Within a PBBN, the encapsulation performed by PIPs also encapsulates connectivity fault management frames sourced by customers attached to CNPs. Encapsulation of S-VLAN and C-VLAN CFM frames hides them from the PBBN enabling a new set of independent MD levels to be defined in the PBBN for connectivity fault management, which may be used for managing B-VLANs spanning between the B-components and BCBs of the PBBN, and over LANs that span between PBB ports. In addition, the backbone may support Maintenance Associations (MAs) per backbone service instance that are identified by

I-SIDs and span between the PIPs and CBPs of I-components and B-components of the PBBN. A full set of eight maintenance levels exists within each PBBN for use by B-VLAN CFM frames. The backbone service instance CFM frames may extend over a E-NNI, therefore providing eight levels of backbone service instance CFM for the extended backbone, which includes all peer connected PBBNs. All eight levels of CFM frames generated in customer networks are carried over the backbone as encapsulated data and may be used by customer networks. Another eight maintenance levels exist for the LANs for which no frames are untagged. This may be the case for the PNP-PNP LAN links. The CFM stacks for the PIP and CBP are depicted in Figure 26-2. The stacks for the CNP and PNP are as depicted in 22.1.1.

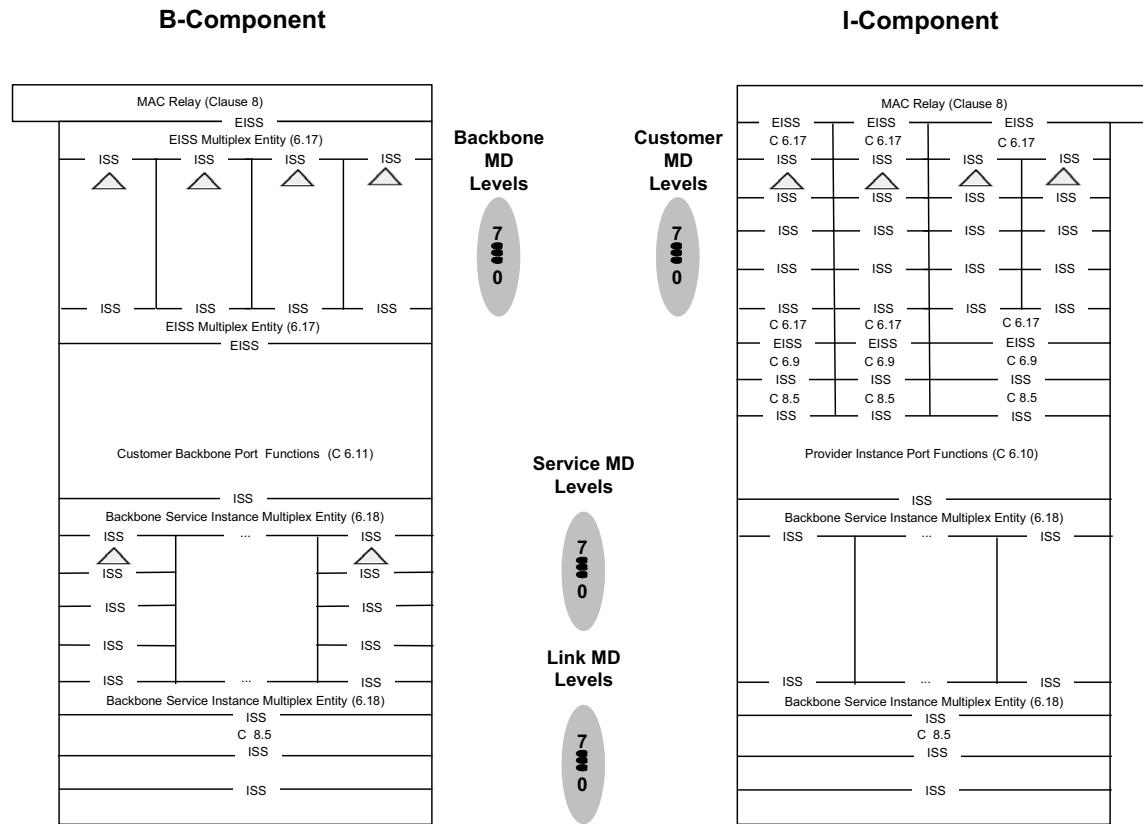


Figure 26-2—CFM shim model

Figure 26-3 and Figure 26-4 show possible MAs at the three types of PBBN service interfaces described in 25.3, 25.4, and 25.5, respectively. These three interfaces are depicted in Figure 26-1, where they are labeled 1, 2, and 3. Figure 26-5 and Figure 26-6 show possible MAs at the two types of PBBN E-NNI interfaces, which are described in 26.6.1 and 26.6.2, respectively. These two types of E-NNIs are depicted in Figure 26-1, where they are labeled 4 and 5. The MAs indicated in Figure 26-3 through Figure 26-6 are example MAs for each PBBN service interface type and PBBN E-NNI type. Typically not all these MAs would be supported within any given PBBN since some MAs illustrated provide redundant capabilities and since the selection of MAs will depend on the service level agreements used for a particular service, on the operational environment of a particular PBBN and on the equipment used in the particular PBBN.

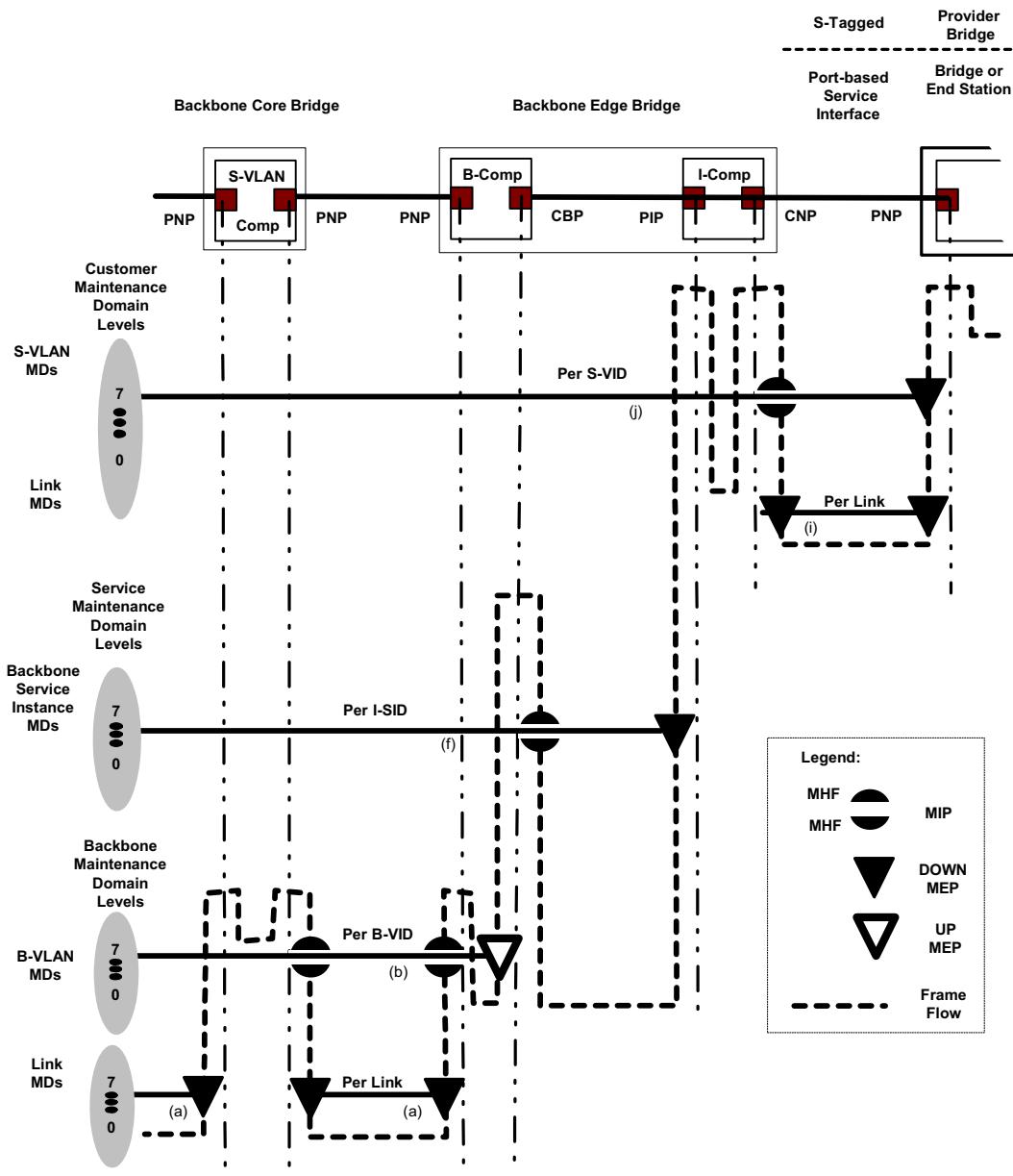


Figure 26-3—CFM example applied to a Port-based and S-tagged Service Interface

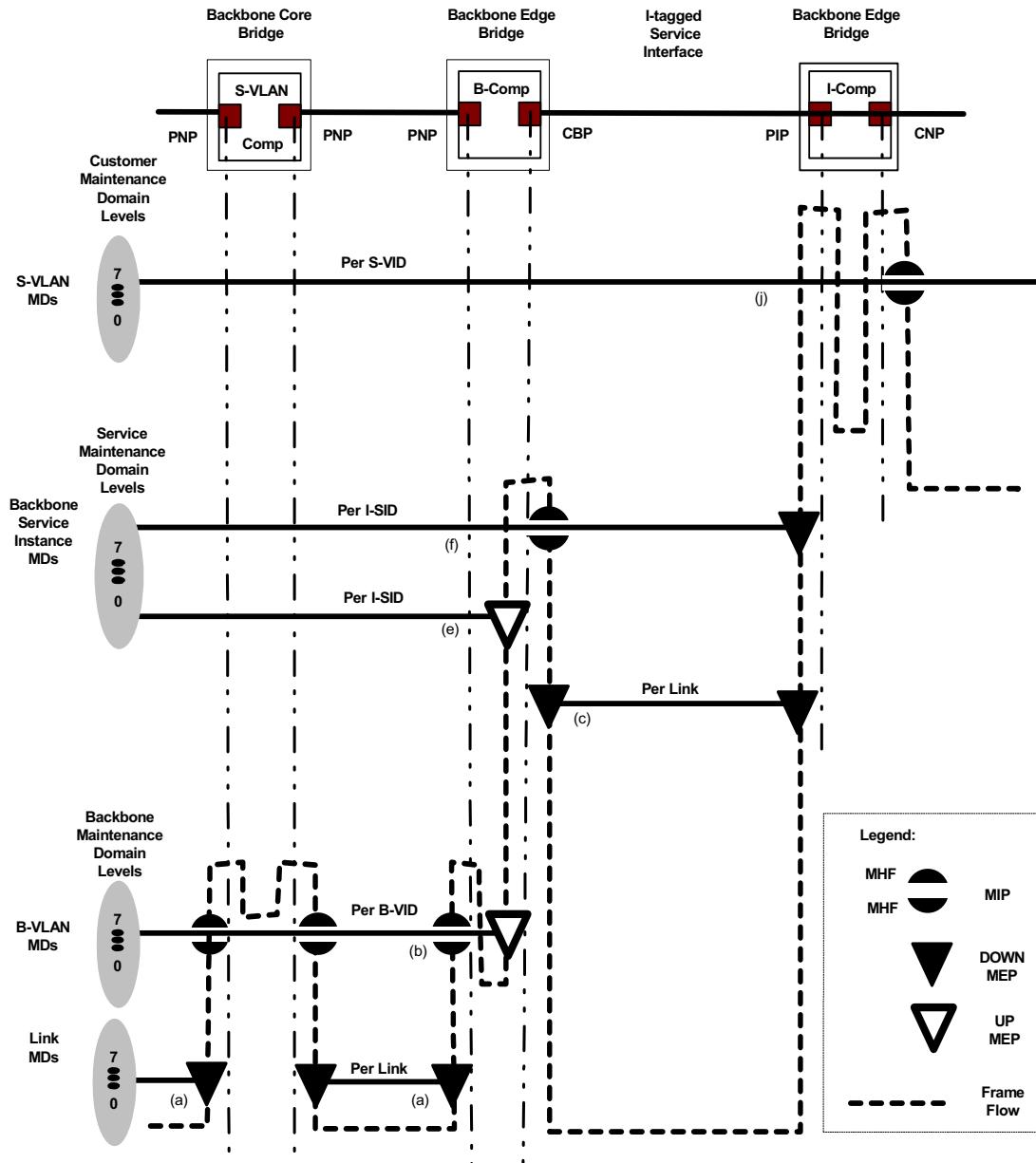


Figure 26-4—CFM example applied to an I-tagged Service Interface

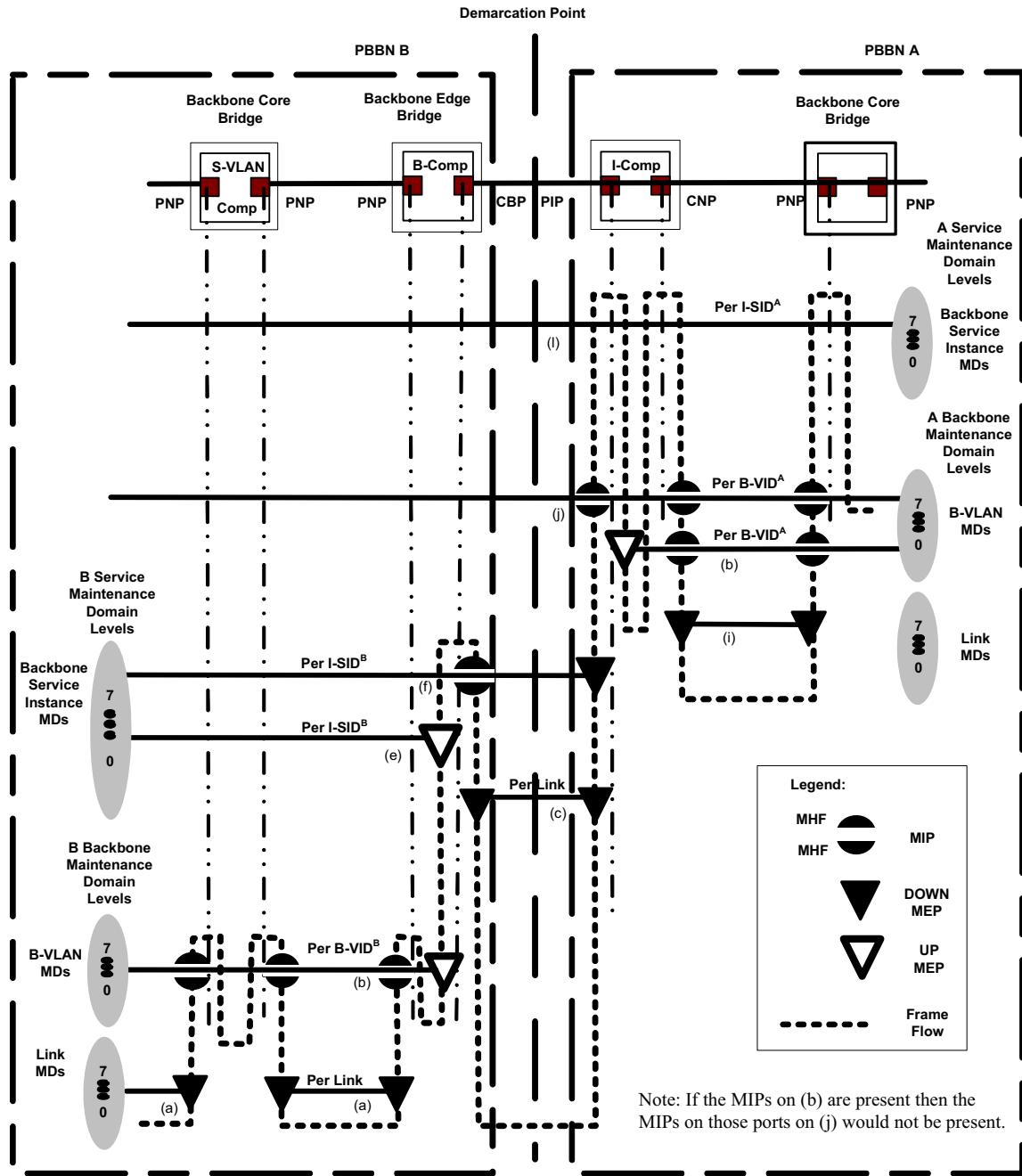


Figure 26-5—CFM example applied to a hierachal E-NNI, CBP-PIP Demarc

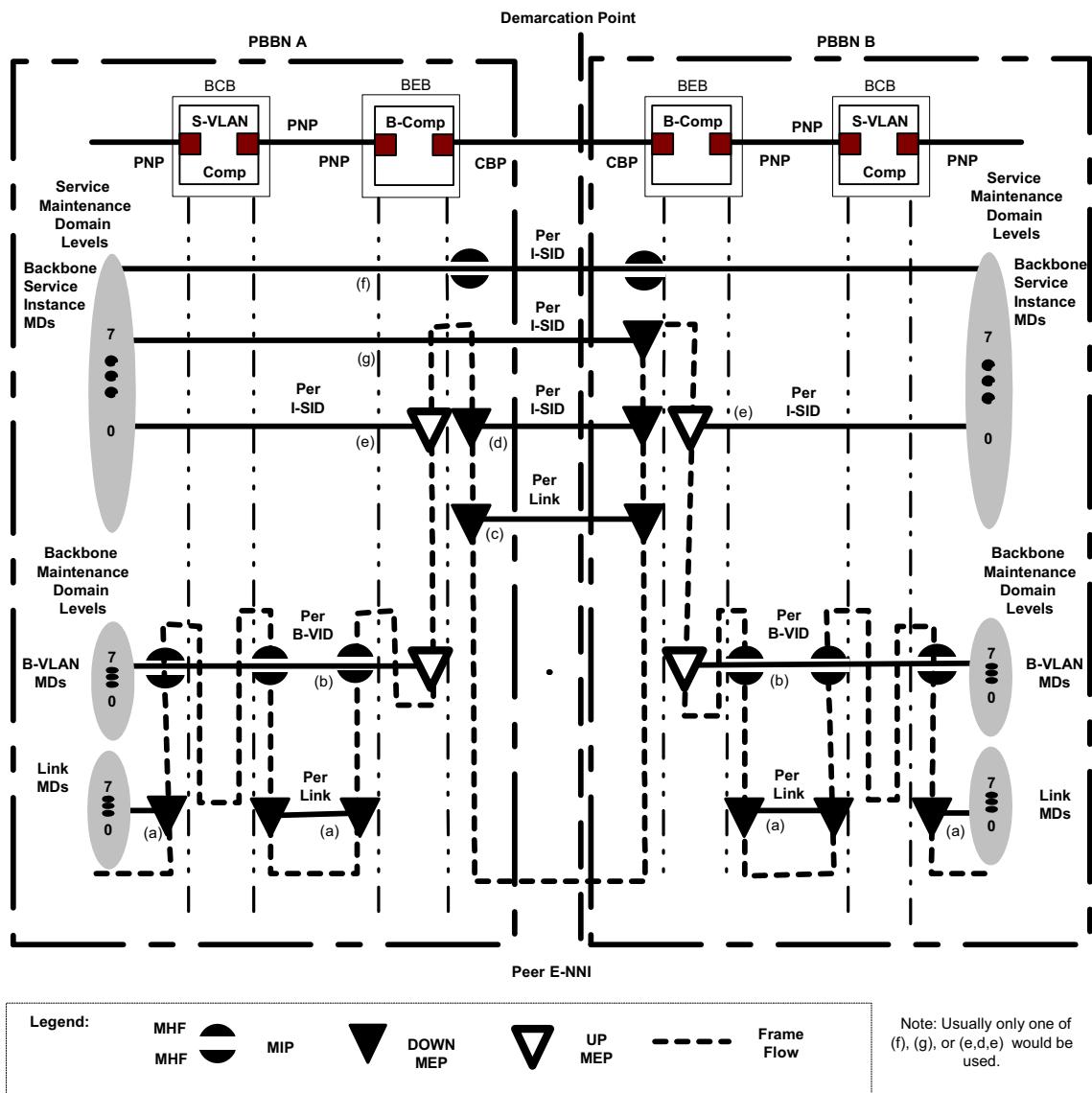


Figure 26-6—CFM example applied to a peer E-NNI, CBP-PIP

In these figures, four distinct categories of MAs are depicted. These are the S-VLAN and C-VLAN MAs, the backbone service instance MAs, the B-VLAN MAs, and the LAN MAs. Each of these four classes of MAs is associated with independent sets of MDs introducing four independent sets of eight maintenance levels. The selection of which levels for each of these MDs to use in a particular network depends on the specific network implementation. The S-VLAN MAs are only visible within the I-component where customer frames are un-encapsulated. Customer CFM frames are encapsulated along with the other customer frames at the VIPs within the I-component. Once the S-VLAN CFM frames are encapsulated, they appear just like any data frame within the PBBN; therefore, they do not activate any CFM functions within the PBBN past the VIPs. The backbone service instance CFM frames may optionally be generated at the Backbone Service Instance Multiplex Entity within the PIP or CBP. These CFM frames are only visible within the PBBN where the I-TAG is being processed, that is at the CBP and at the PIP. Each backbone service

instance CFM maintenance domain may extend over E-NNI boundaries, these eight levels extend over all interconnected PBBNs until they are terminated at a VIP. B-VLAN MAs manage the B-VLANs within a single PBBN. The B-VLAN MDs have a new set of eight maintenance levels, which are separate for each PBBN and never extend out of the PBBN. Finally, the LAN links within a PBBN (PIP to CBP, PNP to PNP), between two PBBNs (CBP to CBP, PIP to CBP) and entering/egressing the PBBN (to/from CNP) may optionally be monitored by LAN link MAs, generated/terminated within the PIP, CBP, PNP, and CNP. LAN link MAs are typically confined to the LAN link.

26.8.1 CFM over Port-based and S-tagged Service Interfaces

Figure 26-3 shows example MAs for the Port-based or S-tagged service interface described in 25.3 and 25.4 and depicted in Figure 26-1 labeled 1 and 2.

MAs (a) and (b) illustrated in Figure 26-3 through Figure 26-6 are used to monitor the PBBN core. Each PBBN has an independent backbone MD level space. MAs (a) and (b) are assigned Backbone Maintenance Levels from the backbone MD level space of their PBBN in the same manner used to assign S-VLAN MAs described in Clause 22. The MAs (a) are used to monitor each individual LAN that interconnects PNPs. The MAs (b) monitor specific B-VLANs. The B-VLAN monitoring, illustrated in the example, begins at the up MEP located at the top of the CBP. A given B-VLAN may exit a B-component on multiple PNPs, each of which may have a MIP on MA (b) if desired.

MAs (i) and (j) illustrated in Figure 26-3 through Figure 26-6 are two possible customer MAs. Attached customers have an independent customer MD level space. MAs (i) and (j) are assigned Customer Maintenance Domain Levels from their customer MD level space in the manner used for S-VLAN MAs described in Clause 22. The MAs (i) monitor the links between the customer equipment and the CNP or the PBBN. At the Port-based or the S-tagged service interface, MAs (i) may be terminated by the provider by an MEP at the CNP and used to monitor the customer's access link. The MAs (j) monitor the service instance from the customer over the PBBN. MAs (j) extend between all the customer attachments of the extended PBBN, passing through any PBBN E-NNI interfaces. At the VIPs located at the edge of the extended PBBN, the CFM frames for MA(j) will be mapped to/from an I-SID and carried over the extended PBBN as encapsulated frames. These CFM frames are encapsulated by the PBBN in the same manner as data frames. The encapsulation makes them invisible to any MPs within the extended PBBN.

In the case of a Port-based service interface, the MA(j) CFM frames are untagged or C-tagged over the interface formed between the CNP and the customer equipment. In this case, the MA(j) CFM frames have an S-VID assigned from the PVID at the CNP, which will be carried to the VIP within the I-comp. In the case of an S-tagged service interface, the MA(j) CFM frames are untagged, C-tagged, or S-tagged over the interface formed between the CNP and the customer equipment. In this case, the MA(j) CFM frames have the S-VID carried over the service interface or, if the CFM frame is not S-tagged, it will have an S-VID assigned from the PVID of the CNP.

MA (f) depicted in Figure 26-3 through Figure 26-6 is used for backbone service instance management. MAs (f) are within the backbone service instance MD level space, which is shared over the entire extended PBBN. MAs (f) are assigned Service Maintenance Domain Levels from this backbone service instance MD level space. The MEPs for MA(f) are formed at the PIP within the Backbone Service Instance Multiplex Entity (6.18) as seen in Figure 26-2. MA(f) may have MIPs located in the CBP also formed by a Backbone Service Instance Multiplex Entity as depicted in Figure 26-2.

26.8.2 Connectivity Fault Management over I-tagged Service Interfaces

Figure 26-4 shows possible MAs for the I-tagged service interface described in 25.5 and depicted in Figure 26-1 labeled 3. The MAs (a) and (b) in Figure 26-4 are used for PBBN core monitoring as described in 26.8.1.

MAs (c) in Figure 26-4, Figure 26-5, and Figure 26-6 are used to monitor the link between the PIP and CBP. The MAs (c) CFM frames are unique among the other CFM frames because they are transmitted without an I-TAG over the CBP to PIP LAN.

MAs (f) depicted in Figure 26-4 are used for service management as described in 26.8.1. In an I-tagged service interface, MAs (f) have an MEP in the customer network located at the Backbone Service Instance Multiplex Entities (6.18) of the customer PIP and may have MIP within the extended provider network located at the Backbone Service Instance Multiplex Entities of the CBPs. The provider may use MAs (e) to monitor each service instance by placing a MEP at the Backbone Service Instance Multiplex Entities of the CBP. The backbone service instance level space is shared between the customer and provider who must agree on the levels used for provider and customer backbone service instance CFM messages.

Customer S-VLAN MAs (j) are encapsulated by the customer equipment and are indistinguishable from data within the PBBN.

26.8.3 Connectivity Fault Management over hierachal E-NNI

Figure 26-5 shows possible MAs for the hierachal E-NNI described in 26.6.1. In Figure 26-1 the hierachal E-NNI labeled 4 depicts an E-NNI with the demarcation point between the CBP and PIP. Example MAs for a hierachal E-NNI with demarcation between CBP and PIP are illustrated in Figure 26-5. The MAs (a) and (b) in Figure 26-5 are used for PBBN core monitoring as described in 26.8.1.

The operation of the backbone service instance and LAN MAs (f), (e), and (c) are as described in 26.8.2. Since each PBBN-A B-VLAN MA (j) is mapped into a PBBN-B backbone service instance monitored by MAs (f) and/or (e) the MAs (f) and (e) may be used to monitor the extended B-VLAN from PBBN-A.

MAs (j) are not visible to PBBN-B as described in 26.8.2. The per PBBN-A backbone service instance MAs (l) illustrated in Figure 26-5 are never visible to PBBN-B since they are encapsulated along with the B-VLAN.

26.8.4 Connectivity Fault Management over peer E-NNI

Figure 26-6 shows possible MAs for the peer E-NNI described in 26.6.2 and depicted in Figure 26-1 labeled 5. The MAs (a) and (b) in Figure 26-6 are used for PBBN core monitoring as described in 26.8.1. The LAN MAs (c) and backbone service instance MAs (e) and (f) in Figure 26-6 are as described in 26.8.2.

The MAs (f), (g), (d), and (c) extend over the demarcation point between PBBN-A and PBBN-B, while the MAs (e) are within PBBN-A and PBBN-B and do not extend over the demarcation point. The MA (f) is a backbone service instance MA that extends through both PBBN-A and PBBN-B providing monitoring for the service from VIP to VIP. MAs (f) may have a MIP on each side of the demarcation point to allow isolation of a fault to an individual PBBN. The MAs (g) also are backbone service instance MAs that extend over the demarcation point. MAs (g) are launched by PBBN-B at the CBP facing the E-NNI rather than at the service originating VIP and extend to the far edge of PBBN-A. MAs (g) may be used by PBBN-B to monitor the backbone service instance over PBBN-A, but only provides information to PBBN-A on the operation of the backbone service instance over the demarcation point to PBBN-B. The MA (d) is a backbone service instance MA that monitors the service over the demarcation point, but does not provide any monitoring deeper in PBBN-A or PBBN-B. MAs (e) provide internal monitoring of the backbone service instance within PBBN-A and PBBN-B, respectively.

26.9 Connectivity Fault Management in a PBB-TE Region

Connectivity Fault Management (CFM) as specified in Clause 18 through Clause 22, provides capabilities useful in detecting, isolating, and reporting connectivity faults in VID-based service instances, backbone service instances, and TESIs. As the original CFM protocol specified in IEEE Std 802.1ag-2007 was primarily focused on VID-based services, the list of additions that are specifically related to TESIs is summarized here.

In particular, this subclause summarizes PBB-TE considerations related to the following:

- a) Addressing PBB-TE MEPs (26.9.1)
- b) TESI identification (26.9.2)
- c) PBB-TE MEP placement in a Bridge Port (26.9.3)
- d) PBB-TE MIP placement in a Bridge Port (26.9.4)
- e) TESI Maintenance Domains (26.9.5)
- f) PBB-TE enhancements of the CFM protocols (26.9.6)

26.9.1 Addressing PBB-TE MEPs

The configuration of a Maintenance Association (MA) requires a parameter (or a list of parameters) that associates it with the monitored service. A TESI is identified by the list of ESPs supporting it. Correspondingly, the configuration of a PBB-TE MA, requires the list of the monitored TESI's ESPs, each identified by the 3-tuple <ESP-DA, ESP-SA, ESP-VID> or collectively by the TE-SID (12.14.5.3.2).

The MEPs related to a TESI require the same set of parameters as a VID-based MEP requires, with the following changes:

- a) The Primary VID is not writable but is always set to the value of the ESP-VID parameter that corresponds to the MA's ESP that has the MEP's MAC address in its ESP-SA field [item d) in 12.14.7.1.3, 20.9.7];
- b) The MAC address of the MEP is the MAC address of the CBP upon which the MEP is operating [item i) in 12.14.7.1.3].

26.9.2 TESI identification

In contrast to VID-based services where the *vlan_identifier* is used to identify the service, independent ESPs could have the same VID value in their ESP-VID field if some other parameters in their identifying 3-tuples are different. Figure 26-7 depicts such a case where the two ESPs are distinguished by their source addresses. The TESI Multiplex Entity (6.19) allows shims defined for PBB-TE to be instantiated per TESI at a Service Access Point that supports multiple TESIs.

26.9.3 PBB-TE MEP placement in a Bridge Port

PBB-TE MEPs are always Up MEPs and can only be placed on CBPs as these correspond to the demarcation points of the ESPs supporting the PBB-TE service. The PBB-TE MEPs 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) in the ESP-VID space identified by the EISS Multiplex Entity (6.17). Since the ESP MAC addresses can be reused by different ESPs, the PBB-TE MEPs need to be further differentiated by the TESI Multiplex Entity (6.19). In principle, separately for each TESI, there can be from zero to eight Up MEPs, ordered by increasing MD Level, from Frame filtering toward Port filtering, even though not more than one MD Level is expected for PBB-TE MAs. Figure 26-8 depicts an example of PBB-TE MEPs on a CBP.

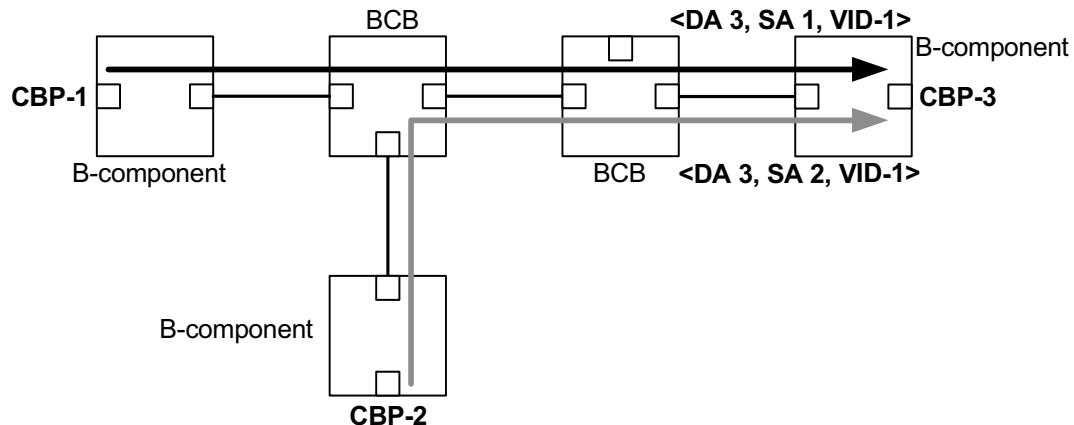


Figure 26-7—Independent ESPs using the same ESP-DAs and ESP-VIDs

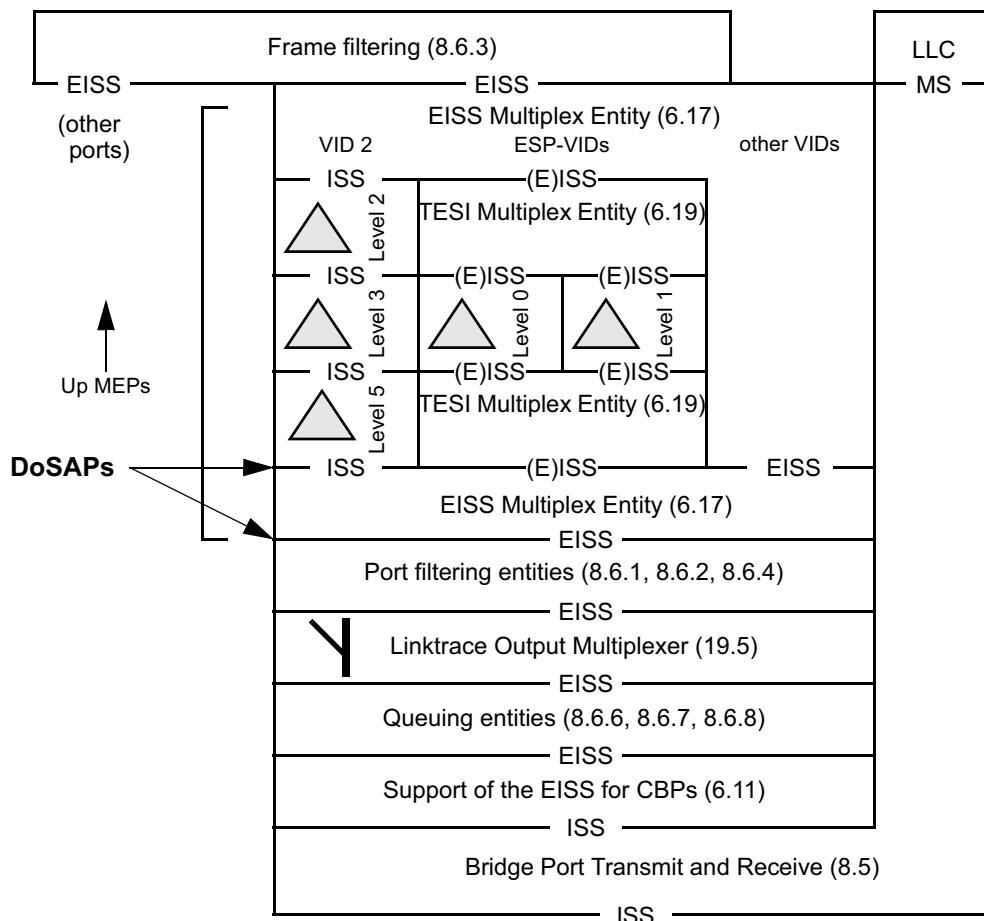


Figure 26-8—PBB-TE MEP placement in a CBP

26.9.4 PBB-TE MIP placement in a Bridge Port

PBB-TE MIPs creation is based on the algorithm described in 22.2.3. PBB-TE MIPs are created per identified ESP-VIDs on PNPs. Figure 22-9 depicts an example of a pair of PBB-TE MHFs on a PNP.

26.9.5 TESI Maintenance Domains

A Provider Backbone Bridge that is capable of providing TESIs supports a new set of TESI MDs in addition to those described in 26.8. In particular the CFM stacks in Figure 26-2 will need to have TESI MD levels in parallel to the depicted Backbone MD levels for CBPs initiating TESIs while TESI MDs will have to be present in parallel to the Backbone MD levels depicted in Figure 26-3.

26.9.6 PBB-TE enhancements of the CFM protocols

26.9.6.1 Continuity Check protocol in a PBB-TE MA

The Continuity Check protocol is described in 20.1 and the corresponding state machines in Clause 20. The only enhancement required by the PBB-TE MA is in the procedure that is responsible for constructing and transmitting a CCM (20.11.1):

- a) The destination_address parameter is set to the MAC address indicated by the value of the ESP-DA field of the ESP having as ESP-SA the MAC address of the MEP emitting the CCM.

PBB-TE MAs may also implement the following enhancements to the Continuity Check protocol:

- b) The procedure MEPprocessEqualCCM() on PBB-TE MEPs does not include the check of the MAID on received CCMs [item b) in 20.17.1].
- c) PBB-TE MEP's may
 - 1) Set the Traffic field (21.6.1.4) bit on transmitted CCMs based on information from the backbone service instance table associated with the CBP upon which the MEP is configured. In particular, the procedure xmitCCM() (20.11.1) is performing the additional action to fill the Traffic field with the presentTraffic variable (20.9.8); and
 - 2) Check the Traffic field bit on received CCMs (20.16.13); and
 - 3) Implement the MEP Mismatch state machines (20.26).

All other Continuity Check processes are the same as those for a VID-based MA.

26.9.6.2 Loopback protocol in a PBB-TE MA

The Loopback protocol is described in 20.2 and the corresponding state machines in Clause 20. The enhancements required by the PBB-TE MA are summarized below:

- a) The LBMs transmitted by a MEP associated with a PBB-TE MA use as the destination_address parameter, the MAC address indicated by the value of the ESP-DA field of the MA's ESP that has the MEP's MAC address indicated in its ESP-SA field [item a) in 20.31.1]. To enable intermediate MIPs to selectively intercept LBMs that are targeting them, the PBB-TE MIP TLV (21.7.5) is inserted as the first TLV [item e) in 20.31.1] in an LBM. The PBB-TE MIP TLV is not included if the LBM destination address in 12.14.7.3.2 is associated with any of the values in the ESP-DA field of the monitored MA's ESPs. The format of the PBB-TE MIP TLV is described in 21.7.5 and is constructed as follows:
 - 1) The MIP MAC address field contains the MAC address of the MIP to which the LBM is targeted [item b) in 12.14.7.3.2];

- 2) The Reverse VID field contains the parameter provided in item f) in 12.14.7.3.2, which it uses as the *vlan_identifier* in the replied LBR;
- 3) The Reverse MAC field is only included if the MEP is associated with a point-to-multipoint TESI. If the MEP is the root MEP, the Reverse MAC field contains the ESP-SA value of any of the point-to-point ESPs in the TESI. If the MEP is a leaf MEP, the Reverse MAC field contains the ESP-DA of the point-to-multipoint ESP in the TESI.
- b) LBMs received by PBB-TE MEPs are not processed and are discarded if the received LBM carries a PBB-TE MIP TLV. A PBB-TE MHF forwards all received LBMs except those carrying a PBB-TE MIP TLV containing in their MIP MAC address field the MAC address of the MIP associated with that PBB-TE MHF [item f) in 20.28.1].
- c) The *destination_address* parameter used by LBRs transmitted by PBB-TE MEPs is the value of the ESP-DA of the MA's ESP that has the MEP's own address in its ESP-SA field. In the case of a PBB-TE MHF, the *destination_address* parameter for the transmitted LBR is
 - 1) The value carried in the Reverse MAC field contained in the PBB-TE MIP TLV of the received LBM, if this is a Group MAC address, or otherwise;
 - 2) The value of the *source_address* of the received LBM.
- d) The *source_address* of LBRs transmitted by PBB-TE MHFs is set to
 - 1) The *destination_address* of the received LBM if this destination address is an Individual MAC address; or otherwise;
 - 2) The value carried in the Reverse MAC field contained in the PBB-TE MIP TLV if the received LBM's destination address is a Group MAC address.
- e) The *vlan_identifier* used by LBRs transmitted by PBB-TE MEPs is set to the value of its Primary VID [item d) in 12.14.7.1.3]. The *vlan_identifier* used by LBRs transmitted by PBB-TE MHFs is set to the value carried in the Reverse VID field contained in the PBB-TE MIP TLV of the received LBM.

All other Loopback processes are the same as those for a VID-based MA.

26.9.6.3 Linktrace protocol in a PBB-TE MA

The Linktrace protocol is described in 20.3 and the corresponding state machines in Clause 20. The enhancements required by the PBB-TE MA are summarized as follows:

- a) The LTMs transmitted by a MEP associated with a PBB-TE MA use as the *destination_address* parameter, the MAC address indicated by the value of the ESP-DA field of the MA's ESP that has the MEP's MAC address indicated in its ESP-SA field [item a) in 20.42.1]. LTMs carry the PBB-TE MIP TLV constructed as follows:
 - 1) The MIP MAC address field is null;
 - 2) The Reverse VID field contains the parameter provided in item e) in 12.14.7.4.2, which is used as the *vlan_identifier* of the replied LTR;
 - 3) The Reverse MAC field is only included if the MEP is associated with a point-to-multipoint TESI. If the MEP is the root MEP, the Reverse MAC field contains the ESP-SA value of any of the point-to-point ESPs in the TESI. If the MEP is a leaf MEP, the Reverse MAC field contains the ESP-DA of the point-to-multipoint ESP in TESI.
- b) No special *destination_address* validation tests are performed by the ProcessLTM() procedure in the case of PBB-TE MAs [item c) in 20.47.1].
- c) The process to identify a possible egress port by an intermediate device that implements a MIP associated with a PBB-TE MA, queries the filtering database of the corresponding bridge, using the *destination_address*, and the *vlan_identifier* of the LTM as the parameters for the lookup (20.47.1.2). The LTMs can be further forwarded by PBB-TE MIPs even if the lookup identifies more than one Egress port.

- d) Finally an LTR issued by a PBB-TE MP
 - 1) Uses as destination_address: the value carried in the Reverse MAC field contained in the PBB-TE MIP TLV of the received LTM, if this is a Group MAC address; or otherwise: the source_address of the received LTM [item a) in 20.47.4].
 - 2) Uses as source_address: its MAC address if the LTR is issued by a PBB-TE MEP; otherwise: the destination_address of the received LTM if this is destination address is an Individual MAC address; otherwise: the value carried in the Reverse MAC field contained in the PBB-TE MIP TLV of the received LTM [item b) in 20.47.4].
 - 3) Sets the vlan_identifier parameter to the value carried in the Reverse VID field contained in the PBB-TE MIP TLV of the received LTM [item c) in 20.47.4].

All other Linktrace processes are the same as those for a VID-based MA.

26.10 Protection switching for point-to-point TESIs

26.10.1 Introduction

In contrast to PBB, spanning tree protocols and broadcasting/flooding are not used in PBB-TE. Filtering databases are populated using a network management system or a control plane, allowing Ethernet Switched Paths (ESPs) to be engineered and provisioned across the network. PBB-TE provides end-to-end linear protection for point-to-point TESIs, where a dedicated protection point-to-point TESI is established for one particular working point-to-point TESI, and the traffic is automatically switched from the working TESI to the protection TESI when a failure occurs on the working entity. The protection entity is preestablished, ensuring the availability of those resources when a defect is detected on the working entity, and allowing for a corresponding sub-50 ms switchover.

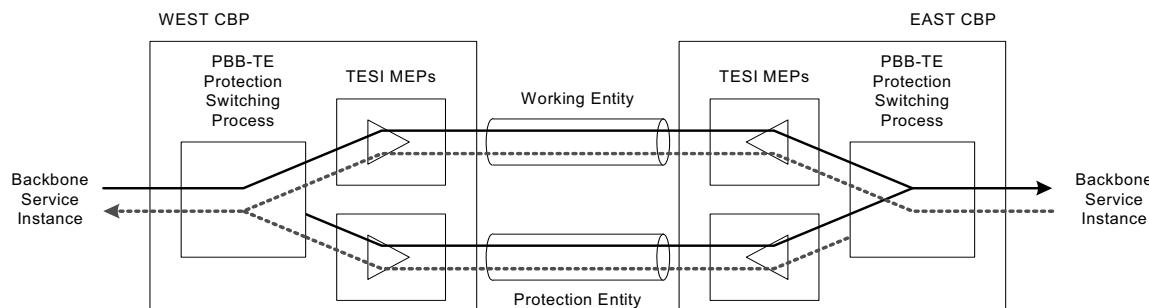


Figure 26-9—Protection switching architecture

Figure 26-9 depicts the essential elements of the PBB-TE protection scheme and its arrangement. In this, 1:1 (i.e., 1 for 1) arrangement, traffic is sent on only one of the paths at any point in time based on information from Operations, Administration and Management (OAM) processes or network operators.

Protection switching may be triggered by manual operation or by CFM information arising from, periodic monitoring of the working and protection paths, or from physical layer monitoring, such as loss of signal or other defects detected through CFM.

The PBB-TE protection switching mechanism aims to offer the capability to switch completely (both ends) in less than 50 ms following the detection of the fault.

NOTE 1—The capability of supporting the above switchover requirement on TESIs, is provided by configuring the interval of Continuity Check Messages on the MAs associated these TESIs to be equal to or lower than 10 ms enabling the detection of the fault and the signalling of the coordination information end to end in less than 50 ms.

The PBB-TE protection scheme may be configurable to be “revertive” or “nonrevertive,” where traffic transmission reverts, or not, to the working path automatically once CFM indicates the fault or defect has been cleared. The protection switching mechanism may also incorporate hold-off and wait to restore timers, the first to allow the fault to be protected by a lower layer protection switching mechanism for instance, while the latter ensures the performance of the working path is fully restored before switching back to it. The hold-off timer obviously slows the overall recovery time for a fault within the protected domain.

NOTE 2—A complete discussion on the various protection switching mechanisms and terminology is provided by ITU-T G.8031 (2009) [B39]. As a caution it should be noted that ITU-T uses the term “bridge” for the logical entity that selects either or both of the transmit paths at the sending end of a protected domain. This is not to be confused with the term Bridge used in this standard.

The description herein, defines and provides a scalable end-to-end resiliency mechanism that offers end-to-end 1:1 bidirectional linear protection switching capable of load sharing for point-to-point TESIs in a PBB-TE Region.

NOTE 3—Some scenarios exist where multiple input events may cause the protection switching state machines in a protection group to select diverse TESIs on which to send traffic. The Mismatch defect informs the operator of such an occurrence thereby allowing the appropriate corrective action to be taken.

Protection for point-to-multipoint TESIs may be provided by external mechanisms or agents, which are out of scope for this specification.

26.10.2 1:1 point-to-point TESI protection switching

In a PBB-TE Region, IB-Backbone Edge Bridges (IB-BEBs) constitute the demarcation between the Provider Backbone Bridged Network (PBBN) of interest and the networks attached to it. The protected domain is defined to be the area between the Customer Backbone Ports (CBPs) on the different B-Components of the involved IB-BEBs.

The starting point in providing protection switching is the creation and configuration of a TE protection group. A TE protection group is a group of two point-to-point TESIs between a pair of CBPs, which carries an assigned set of backbone service instances, and continues to carry these backbone service instances if any one of the TESIs in the group is failed or disabled. The creation of the TE protection group requires the assignment of one TESI as a working entity and a second TESI with the same terminating points as a protection entity providing a different bidirectional connectivity path (12.14.1.2). The configuration of the TE protection group is completed with the assignment of the list of the backbone service instances that are to be protected by the TE protection group [item b) in 12.18.2.1.3].

The protection switching mechanism is capable of load sharing as the TESIs that are assigned to a TE protection group can be reused in a number of TE protection groups enabling a list of I-SIDs to be distributed among a set of interdependent TE protection groups [item c) in 12.18.1.1.3]. A set of interdependent TE protection groups forms a coordinated protection group. Protection switching requests to interdependent TE protection groups must be coordinated for an operator to manage the TESIs in a coherent manner and to avoid potentially competing requests for each TESI. For example, to unconditionally remove traffic from a TESI, one request to Lock Out the TESI results in a request to each TE protection group to which the TESI belongs, either LoP or FS according the role of the TESI in each group. Similarly, to conditionally remove traffic from a TESI, one Manual Switch request for the TESI results in the appropriate Manual Switch request to each TE protection group to which the TESI belongs. Allowing a single operator request per coordinated protection group and coordinating the resulting requests to each TE protection group reduces the operational complexity of the set of interrelated TE protection groups. Using a coordinated protection group, the service traffic (BSI) load can be shared across a set of TESIs and at the same time be protected against single faults by 1:1 protection.

NOTE 1—If the operator initiates switching of an individual load shared TE protection group the mmCCMdefect may not detect a switch mismatch.

NOTE 2—The load sharing mode of operation corresponds to 1:1 ETH Sub Network Connection Group protection with Inherent monitoring (ETH SNCG/I) in the ITU-T protection switching terminology.

The following description of the PBB-TE protection switching mechanism is an example provided for illustrative purposes. Figure 26-10 depicts a network where two point-to-point TESIs have been provisioned. The TE-1 service instance on the top of the figure consists of two ESPs each identified by a 3-tuple: <CBP-B, CBP-A, VID-1> is depicted in black and <CBP-A, CBP-B, VID-2> in gray. The TE-2 service instance below consists of two other ESPs identified by the 3-tuples <CBP-B, CBP-A, VID-3> and <CBP-A, CBP-B, VID-3> which are depicted in white. The use of the same ESP-VID on the TE-2 ESPs is only for illustrative purposes. There are no explicit requirements on using the same or different ESP-VIDs on the ESPs comprising the TESIs that are assigned to a TE protection group. One set of backbone service instances, shown by the black arrows just outside the two CBPs, has TE-1 assigned as its working TESI and TE-2 assigned as its protecting TESI.

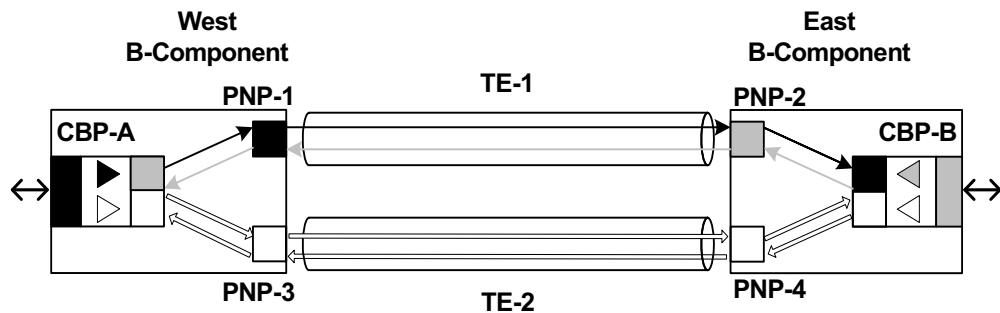


Figure 26-10—PBB-TE point-to-point protection switching

The ESPs are provided by configuring entries in the Filtering Databases on all the Bridges that these ESPs need to pass through. On the terminating IB-BEBs, the VLAN membership of each participating port has to be configured for the B-components. In Figure 26-10, this is depicted by the color of the boxes that are placed inside each of the B-Component boxes. For example, the upper right PNP-1 port on the West B-component is a member of the VID-1 (black), while the CBP on the same component is a member of VID-2 (gray) and VID-3 (white). Only frames tagged with the appropriate VID parameter can egress these ports.

NOTE 3—The PNPs are shown for illustrative purposes only and that there is no explicit requirement to diversify PNPs on the terminating IB-BEBs.

Each of the TESIs is monitored by an independent Maintenance Association. One MA is set to monitor the top (TE-1) TESI and a second to monitor the bottom (TE-2) TESI. Each of these two MAs is associated with a pair of ESPs and identified by their corresponding 3-tuples. Two Up MEPs, associated with the MA monitoring the TE-1 service instance, are configured on the CBPs that terminate the TE-1 service instance. Each of these MEPs has its own primary VID, VID-1 (black) for the MEP on the West B-component associated with the TE-1 service instance, and VID-2 (gray) for the MEP on the East B-component. In this configuration each MEP receives frames that are tagged with the ESP-VID corresponding to the primary VID of the associated remote MEP and can only send frames that are tagged with its own primary VID. In particular, in the depicted example the MEP for the top entity on the West B-component can send only CCMs tagged with VID-1 (black) and receive CCMs tagged with VID-2 (gray), while the corresponding MEP on the East component can only send CCMs tagged with VID-2 (gray) and receive CCMs tagged with VID-1 (black). The primary VID of each MEP is depicted by the color of the MEP. It should be also noted that all the depicted ports in Figure 26-10 have their Enable Ingress Filtering parameter (8.6.2) reset to its default value, i.e., Disable Ingress Filtering.

Data traffic is mapped to a TESI by configuring the CBP parameters (6.11). In particular the CBP backbone service instance identifier is used to allow specific service instances to be carried by the TESI while the entries of the B-VID column in the backbone service instance table are indicating how to map the identified service instances to a specific ESP. The CBP's B-VID value is depicted in Figure 26-10 by the color of the vertical bars (black for CBP-A, gray for CBP-B) placed at the tips of the arrows representing the set of backbone service instances.

As a result customer frames that need to be transported by a TESI and are identified by a specific I-SID and reach the CBP on the West B-component from the left can be mapped to the black ESP or the white ESP while frames on the same service that reach the CBP on the East B-component from the right will be mapped to the gray ESP or the white ESP based on the CBP's configured B-VID value. If all the services are normally mapped to the black ESP and gray ESP, then TE-1 would correspond to the working entity and TE-2 to a stand-by protection entity. Irrespective of how the data traffic is mapped to the TESIs, CCM frames are exchanged on both TESIs in order to regularly check the provided connectivity.

If a fault occurs on any of the ESPs, the MEP on the receiving end will be notified. In particular if a fault on the black ESP occurs, as shown in Figure 26-11, the MEP on the East B-component will declare a remote MEP defect by setting the rMEPCCMdefect parameter (20.19.1). The timer counter for timing out CCMs has a granularity finer than or equal to 1/4 of the time represented by the CCMinterval variable (20.8.1). A Bridge does not set rMEPCCMdefect within $(3.25 \times \text{CCMinterval})$ seconds of the receipt of a CCM, and sets rMEPCCMdefect within $(3.5 \times \text{CCMinterval})$ seconds after the receipt of the last CCM. The setting of the rMEPCCMdefect parameter will result in a change of the appropriate B-VID entries in the backbone service instance table, of the east CBP from the gray VID-2 to the white VID-3, which is the ESP-VID of the associated ESP on the protection TESI.

NOTE 4—The B-VID parameter will also change when the xConCCMdefect (20.23.3) or the errorCCMdefect (20.21.3) parameters are set as these indicate a very serious misconfiguration problem.

All subsequent CCMs sent by the east gray MEP on TE-1 will have the RDI field set for as long as proper CCMs are not received by the MEP. A reception of a CCM frame with the RDI field set (or an event that causes setting of the someRMEPCCMdefect, xConCCMdefect or errorCCMdefect) by the MEP associated with TE-1 on CBP-A will cause a change of the appropriate B-VID entries in the backbone service instance table of the CBP-A on the West B-component, to the preconfigured value of the protection ESP. The result will be to move the affected service instances to the protection TE-2 service instance as depicted Figure 26-11. Consequently, the approximate maximum time before a bidirectional switch occurs following a unidirectional failure would be approximately equal to $(3.5 \times \text{CCMinterval}) +$ the time required for a CCM PDU to travel between the MEPs monitoring the TESI + the time to update the associated backbone service instance table entries.

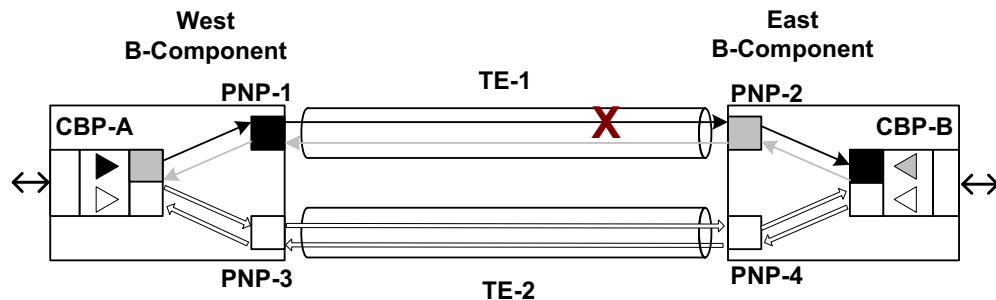
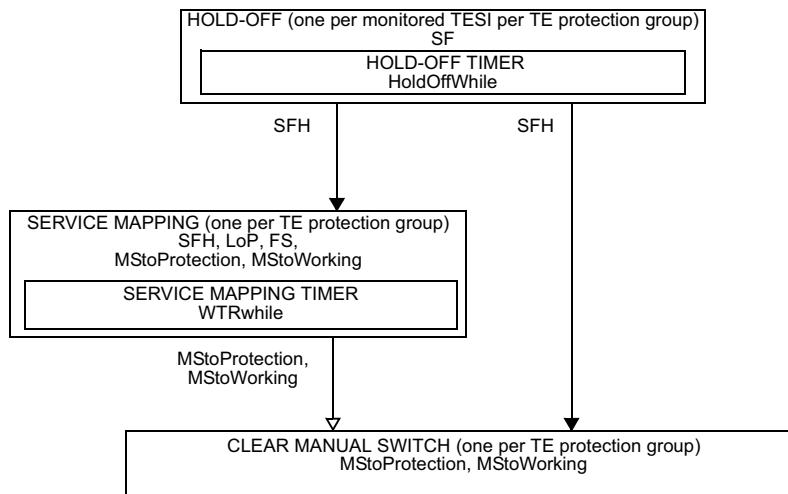


Figure 26-11—Mapping data traffic to the protection entity

26.10.3 Protection Switching state machines

The local protection commands and protection behavior specified here follow the architectural model used in ITU-T G.8031 (2009) [B39]. The relationships among the protection switching state machines are illustrated in Figure 26-12. That figure uses conventions similar to those of 13.22, and of Figure 13-13: 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.


NOTATION:

A variable is shown within the machine that uses it, or between machines with an arrow head style that indicates how it is used to communicate:

→ Not changed by the target machine.

→ Set (or cleared) by the originating machine, cleared (or set) by the target machine.

Figure 26-12—Relationships of the Protection switching state machines—overview

There is one set of Protection Switching state machines per TE protection group per CBP. The TE protection group consists of the working and the protection TESIs. The Protection Switching state machine reuses the defect variables that are presented in Table 20-1. Variables and procedures that are preceded with “w.” refer to the working entity while those that are preceded with “p.” refer to the protection entity. These internal prefix/entity associations remain the same following a protection switch to the protection entity.

26.10.3.1 Notational conventions used in state diagrams

The Protection Switching state machines are specified using the notational conventions defined in Annex E.

26.10.3.2 State machine timers

The timer variables declared in this subclause are part of the specification of the operation of Protection Switching. 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 may be implemented per Service Mapping state machine:

- a) WTRwhile (26.10.3.2.1)

One instance of the following may be implemented per Hold-off state machine:

- b) HoldOffWhile (26.10.3.2.2)

26.10.3.2.1 WTRwhile

A timer to be used to prevent frequent operation of the protection switch due to an intermittent defect. This timer allows for a fixed period of time to elapse before data traffic is mapped from the protection TESI to the working TESI when in revertive mode.

26.10.3.2.2 HoldOffWhile

In order to coordinate timing of protection switches at multiple layers or across cascaded protected domains, a hold-off timer may be required. The purpose is either to allow a server layer protection switch to have a chance to fix the problem before switching at a client layer, or to allow an upstream protected domain to switch before a downstream domain.

26.10.3.3 Protection Switching variables

The following variables are defined for the Protection Switching state machines:

- a) BEGIN (26.10.3.3.1)
- b) SF (26.10.3.3.2)
- c) SFH (26.10.3.3.3)
- d) LoP (26.10.3.3.4)
- e) FS (26.10.3.3.5)
- f) MStoProtection (26.10.3.3.6)
- g) MStoWorking (26.10.3.3.7)
- h) WTRTime (26.10.3.3.8)
- i) HoldOffTime (26.10.3.3.9)

Table 26-8 illustrates the inherent priority of each protection request.

Table 26-8—Protection Requests Hierarchy

Priority	Request
highest	LoP
	FS
	p.SF
	w.SF
	MStoProtection, MStoWorking
	WTR
lowest	NoRequest

NOTE—The priorities associated with the various requests in this table are in general alignment with the corresponding priorities in ITU-T G.8031 (2009) [B39]. The only difference is that the precedence of p.SF and FS are inverted since ITU-T G.8031 relies on an APS protocol to be running on the protection path.

26.10.3.3.1 BEGIN

This is a Boolean variable controlled by the system initialization process. A value of TRUE causes all Protection Switching state machines per TE protection group to continuously execute their initial state. A value of FALSE allows these state machines to perform transitions out of their initial state, in accordance with the relevant state machine definitions.

26.10.3.3.2 SF

SF is the logical OR of someRMEPCCMdefect (20.35.5), someRDIdefect (20.35.7), xConCCMdefect (20.23.3), and errorCCMdefect (20.21.3).

NOTE—Locally detected MAC/PHY faults may also be components of SF (e.g., to enable potentially faster fault detection), but are outside the scope of this standard.

26.10.3.3.3 SFH

A Boolean flag set and cleared by the Hold-off state machine to indicate that SF is set for a period which is equal to, or larger than, the HoldOffTime. The variable will be set equal to SF if the Hold-Off timer is not supported.

26.10.3.3.4 LoP

A Boolean flag indicating the administrative state of the protection entity. If set, it prohibits the use of the protection entity. Its value can only be controlled by an administrator action [item b2) in 12.18.2.3.2].

26.10.3.3.5 FS

A Boolean flag indicating the presence of an administrative command to force switch the data traffic to the protection TESI. Its value is set to 1 by an administrator action [item b3) in 12.18.2.3.2]. It can be reset by an administrator action that corresponds to a request of the same or higher priority according to Table 26-8.

26.10.3.3.6 MStoProtection

A Boolean flag indicating the presence of an administrative command to manually switch the data traffic to the protection TESI, in the absence of a failure of the working or the protection TESI. Its value is set to 1 by an administrator action [item b4) in 12.18.2.3.2]. It can be reset by an administrator action that corresponds to a request of the same or higher priority according to Table 26-8 and by the operation of the Clear Manual Switch state machine.

26.10.3.3.7 MStoWorking

A Boolean flag indicating the presence of an administrative command to manually switch the data traffic to the working TESI in the absence of a failure of the working or the protection TESI. Its value is set to 1 by an administrator action [item b5) in 12.18.2.3.2]. It can be reset by an administrator action that corresponds to a request of the same or higher priority according to Table 26-8 and by the operation of the Clear Manual Switch state machine.

26.10.3.3.8 WTRTime

The wait-to-restore (WTR) period, as provided by the corresponding managed object [item c) in 12.14.6.1.3]. May be configured by the operator in 1 min steps between 5 and 12 min; the default value is 5 min. A value of 0 indicates nonrevertive mode. The overall accuracy of the WTR timer (e.g., <±25 ms) should be sufficient to allow both CBPs to revert back to the working entity in less than 50 ms.

26.10.3.3.9 HoldOffTime

The Hold-Off period as provided by the corresponding managed object [item f) in 12.14.6.1.3]. The suggested range of the hold-off timer is 0 to 10 s in steps of 100 ms (accuracy of ± 5 ms); the default value is 0.

26.10.3.4 Protection Switching procedures

The following procedures are defined for the Service Mapping state machine:

- a) mapDataToWorking() (26.10.3.4.1)
- b) mapDataToProtection() (26.10.3.4.2)

NOTE—Since a merging selector is used, there is the potential for mis-ordered frames at egress due to differential delay between the working and protection TESIs following a protection switch. If this is a concern, then briefly discarding frames prior to executing either mapDataToX() processes described in 26.10.3.4.1 or 26.10.3.4.2 will avoid the issue (e.g., 1 ms is equivalent to a 20% differential delay across 1000 km of fiber).

26.10.3.4.1 mapDataToWorking()

Maps the customer service(s) that are to be transported by a TE protection group [item b) in 12.18.2.1.3], to the working TESI identified by the 3-tuples <remote CBP MAC, local CBP MAC, ESP-VID(w)>, <local CBP MAC, remote CBP MAC, ESP-VID(w')>, by setting the VID values of the corresponding I-SID entry(-ies) in the backbone service instance table to the ESP-VID(w).

26.10.3.4.2 mapDataToProtection()

Maps the customer service(s) that are to be transported by a TE protection group [item b) in 12.18.2.1.3], to the protection TESI identified by the 3-tuples <remote CBP MAC, local CBP MAC, ESP-VID(p)>, <local CBP MAC, remote CBP MAC, ESP-VID(p')>, by setting the VID values of the corresponding I-SID entry(-ies) in the backbone service instance table to the ESP-VID(p).

26.10.3.5 Protection Switching state machine diagram

The Protection Switching state machines implement the function specified by the state diagrams in Figure 26-13, Figure 26-14, and Figure 26-15, the variables in 26.10.3.3, and the procedures in 26.10.3.4.

The Hold-off state machine shall be present only when the Hold-off timer is supported.

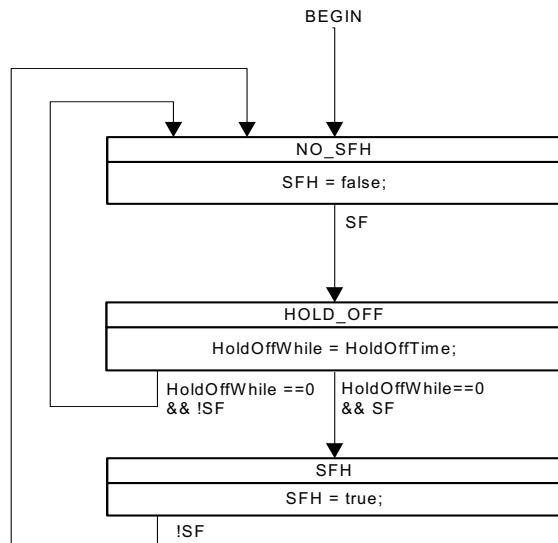


Figure 26-13—Hold-off state machine

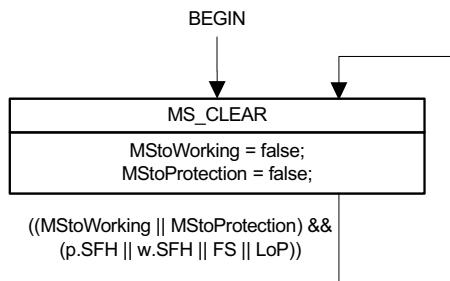


Figure 26-14—Clear Manual Switch state machine

The current state of the Service Mapping state machine is available as a managed object [item c) in 12.18.2.1.3]

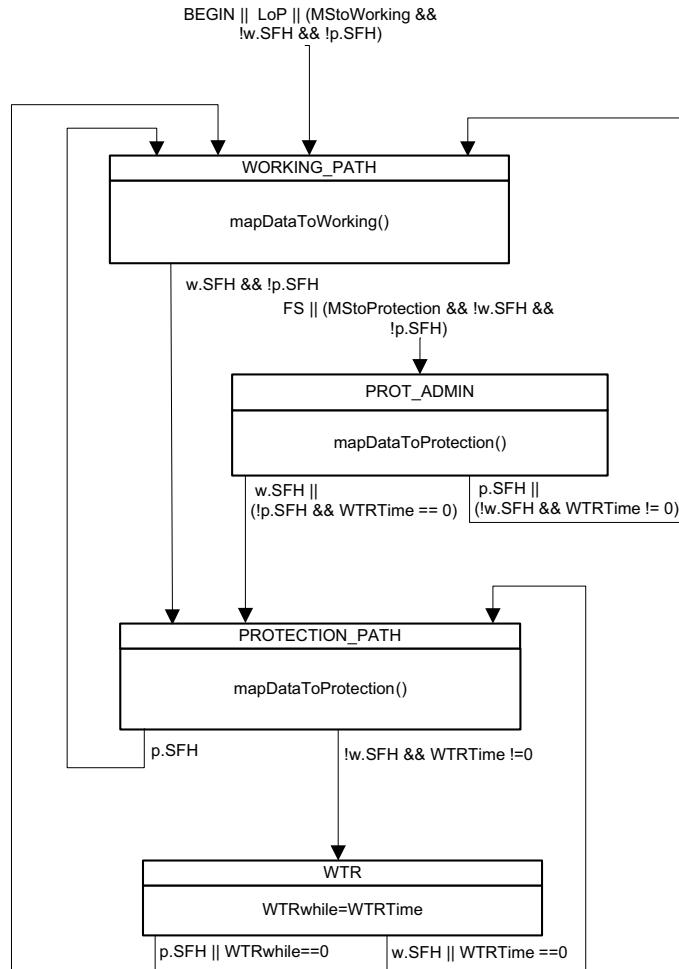


Figure 26-15—Service Mapping state machine

26.11 Mismatch defect

Under certain equipment malfunction conditions and/or wrong configuration, a mismatch between the mappings of the backbone service instances to the appropriate TESI at the terminating CBPs can happen. To maintain the proper operation of the network, this mismatch should be detected and reported. There can be two types of mismatch. They are as follows:

- Protection switching incomplete
- Working/protection configuration mismatch

An example of protection switching incomplete mismatch is when, due to certain equipment malfunction, the near end (East B-Component in Figure 26-11) fails to switch over but it sends a CCM with the RDI field set to the far end (West B-Component in Figure 26-11). The far end transmits to the protection TESI while the near end is still transmitting in the working TESI. Similarly a mismatch can also happen when the near end transmits to the protection TESI but the far end fails to start transmitting to the protection TESI when it receives a CCM with the RDI field set.

The mismatch can also happen because of wrong configuration. For example, one end is configured to send traffic on working TESI while the other end is configured to send traffic on protection TESI. Or one end is configured as revertive mode while the other end is configured as nonrevertive mode. The mismatch occurs when a failure is cleared.

NOTE—Since a merging selector is used there is not necessarily a traffic loss in all mismatch cases.

PBB-TE MEPs that support the Traffic field (21.6.1.4) can detect this defect. The Mismatch state machine in 20.24 is providing this capability.

27. <Reserved for a future amendment>

This clause has been reserved for use by a future amendment.

28. <Reserved for a future amendment>

This clause has been reserved for use by a future amendment.

29. DDCFM operations and protocols

29.1 Principles of DDCFM operation

Data-driven and data-dependent connectivity fault management (DDCFM) comprises tools to facilitate the diagnosis and isolation of data-driven and data-dependent faults in Virtual Bridged Local Area Networks. This clause describes the functions of DDCFM and how they can be operated and managed. DDCFM is an extension to Connectivity Fault Management specified by Clause 18 through Clause 22. As in the case of CFM, DDCFM capabilities can be used in networks operated by multiple independent organizations, each with restricted management access to others' equipment.

29.1.1 Data-driven and data-dependent faults (DDFs)

There are two broad types of faults in Virtual Bridged Local Area Networks that affect only certain frames or sequences of frames. Simple data-dependent faults are those that result in the repetitive loss or misdirection of each of those frames, independently of any other frames, and are usually the result of simple misconfiguration or of a failure to appreciate the consequences of a configuration option (installing protocol specific filters, for example). Data-driven faults are more complex: the presence (or absence) of some data frames causes or contributes to the loss of other frames. While the services supported by Virtual Bridged Local Area Networks are conceptually data independent, the use of data-driven techniques enables enhanced service delivery. To give three examples: multicast frame filtering and consequent bandwidth saving is facilitated by Internet Group Membership Protocol (IGMP) snooping; stateful firewalls are used to protect users connected to managed services; and efficient allocation of frames to the individual links of an aggregation (IEEE 802.1AX Link Aggregation) is often based on spotting conversations by looking at frame data.

29.1.2 Basic principle to diagnose and isolate DDFs

The basic principle to diagnose DDFs is to isolate them to a small enough segment of the network, such as a Bridge Port, a member port of LAG within a Bridge Port, or a LAN. Once DDFs are isolated, understanding the cause of the fault at this location becomes much easier.

The basic procedure to achieve isolation of a DDF is to divide the network into segments and determine whether certain data frames can traverse through each segment as expected. When a network segment is identified as faulty, the segment is further divided into smaller segments until a bridge, a Bridge Port, or a member port of LAG within a Bridge Port is identified as not passing data frames as expected.

A DDF may not be apparent in the absence of live traffic (that is, when test data are used). DDCFM is designed to carry out the diagnosis while live traffic is being exchanged.

DDCFM facilitates diagnosis and consequent isolation of DDFs with the following two techniques: forward path test (FPT) and return path test (RPT).

29.1.2.1 Forward path test

The goal of forward path test (FPT), as depicted in Figure 29-1, is to determine whether specified data frames (frames associated with a service instance, or data frames with certain destination addresses or VID, etc.) can reach a particular location (which could be a Bridge Port, or a member port of LAG within a Bridge Port) without error. FPT's Reflection Responder (29.2.2) reflects the identified data frames (e.g., a service instance or selected data frames) to a specific target location, which could be an end station, a bridge, or a test device. The data frames to be reflected can also be continued to their original destination(s). This option, called the Continue option throughout this standard, is to support testing of a data stream in a manner transparent to the application sourcing or sinking that data stream.

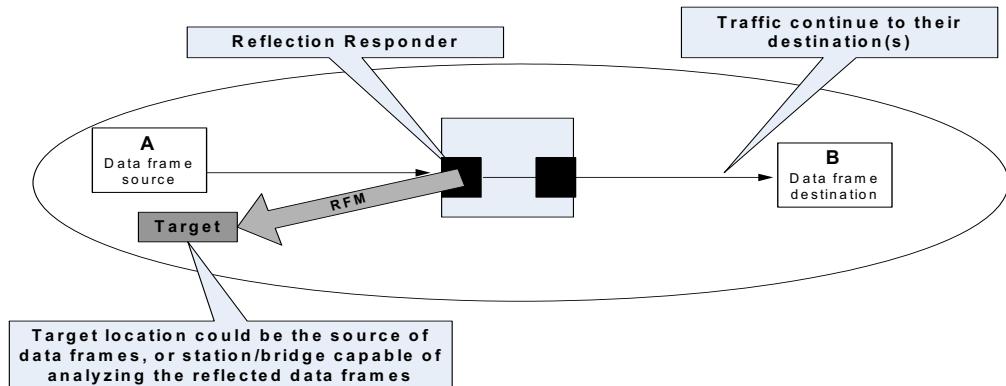


Figure 29-1—Forward path test (FPT)

FPT can be configured on ports where no MPs exist. However, when they are implemented within MPs or configured on ports with MPs, better fault isolation can be achieved. For example, if the target location does not receive the expected reflected data frames in the specified time span, it can send a LBM (21.7) to verify the connectivity between the Target location and the MP on which the Reflection Responder (29.2.2) is configured. For a network that supports MEPs but not MIPs, FPT can be used to diagnose segment connectivity by configuring a Reflection Responder on an intermediate node to encapsulate the received CCM in RFM format (29.4.2) and forward them to a target location.

The receiver at the target location records the reflected frames for examination by the operator and may also transfer the reflected frames to an analysis function. Some examples of target location analyzers are as follows: comparing the reflected frames with the original ones to verify if there are any errors, running a proxy application at the target location to simulate the handshakes as if those packets actually reach their original destinations, etc.

As in CFM (Clause 18 through Clause 22), a network operator can only set or activate FPT's Reflection Responder within its own maintenance domain. At the customer's request, a network provider may use DDCF to verify whether DDFs occur within its domain. FPT's Reflection Responder (29.2.2) can only be created or activated by a network provider's own Network Management System via secure interface. Therefore, customers or other operators cannot set a FPT Reflection Responder (29.2.2) in a maintenance domain that does not belong to them.

In order for intermediate bridges and/or a target location to distinguish the reflected data frames from other traffic, each reflected frame is encapsulated with an RFM header. The RFM header added to the reflected data frame carries the same Maintenance Domain level associated with the Reflection Responder, so that the reflected data frames, each encapsulated with a proper RFM header, are contained within the same Maintenance Domain. The target location has to be in the same maintenance domain to receive the RFMs.

Depending on the conditions specified for selecting data frames to be reflected, some conditions can cause large amount of data frames to be reflected. If there are multiple locations within a network reflecting a large number of data frames simultaneously, excessive traffic could be added to some segments of the network, which may impact network performance. To avoid excessive traffic being added to the network by a Reflection Responder, this standard recommends only one Reflection Responder be activated at a time within one maintenance domain.

FPT consists of Reflection Responder configuration, the action to reflect identified data frames, and the analysis of the reflected data frames. The last step, i.e., analysis of the reflected data frames, is beyond the scope of this standard.

29.1.2.2 Return path test

The return path test (RPT) is to determine whether a flow of data frames can be sent without error from a specific location within a network to a station or stations specified by the destination_address associated with each of the frames under test. The RPT is achieved by an originating station encapsulating each frame of the flow under test with an SFM header, a Decapsulator Responder (29.2.6) removing the SFM header, optionally placing its own MAC in the source_address field of the decapsulated data frames, and sending the data frames to a station or stations as specified by the destination_address field. By default, RPT's Decapsulator Responder places its own MAC address in the source_address field of the decapsulated data frames before sending them out. However, a network operator can force a Decapsulator Responder not to change the source_address field of the decapsulated data frame before sending it out to conduct special purpose testing.

The RPT allows a network operator to inject specific data frames into the network for testing purposes. To prevent one network operator from injecting traffic to other networks, it is necessary for the associated managed object to be created under secure mode and for SFM originator to be in the same Maintenance Domain as the MP on which Decapsulator Responder is defined.

The Figure 29-2 illustrates a simple case of injected traffic from Node B to reach Node A. Multiple data frames can be injected and injected data frames could have different addresses or multicast addresses in their corresponding destination_address field.

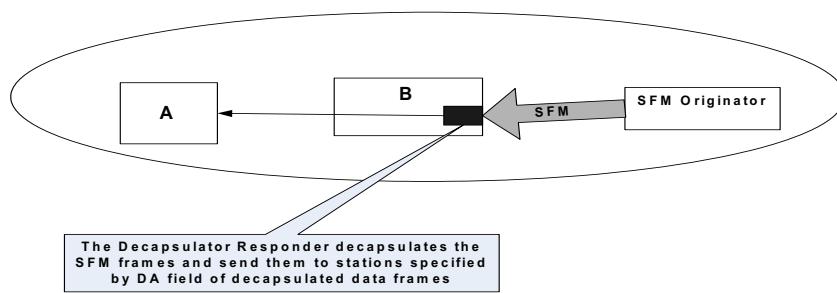


Figure 29-2—Return path test

The RPT consists of the Decapsulator Responder configuration, the sending of SFMs, the action to decapsulate and forward, and the analysis of the received data frames at their destination(s). The last step, i.e., the analysis of the received data frames at their destination(s), is beyond the scope of this standard.

29.1.2.3 Derived testing scenarios

Forward path test and return path test can be used together in various ways to achieve more sophisticated testing to diagnose and isolate data-driven and data-dependent faults. It is beyond the scope of this standard to elaborate different ways of combining forward path test and return path test. This subclause only illustrates one example of using both forward path test and return path test.

Figure 29-3 shows how to achieve segment fault isolation in the middle of a network using FPT and RPT together. By creating a Decapsulator Responder and Reflection Responder on two Bridge Ports in the middle of a network, a network operator can test if the specified data frames can traverse through the segment or not. This type of testing is especially useful to diagnose data-driven and data-dependent faults out of firewall. This type of testing can also be used to diagnose a connectivity fault between two intermediate Bridge Ports (or two MIPs) by setting the Reflection Responder filter (29.2.2.1) to the specified CCMs.

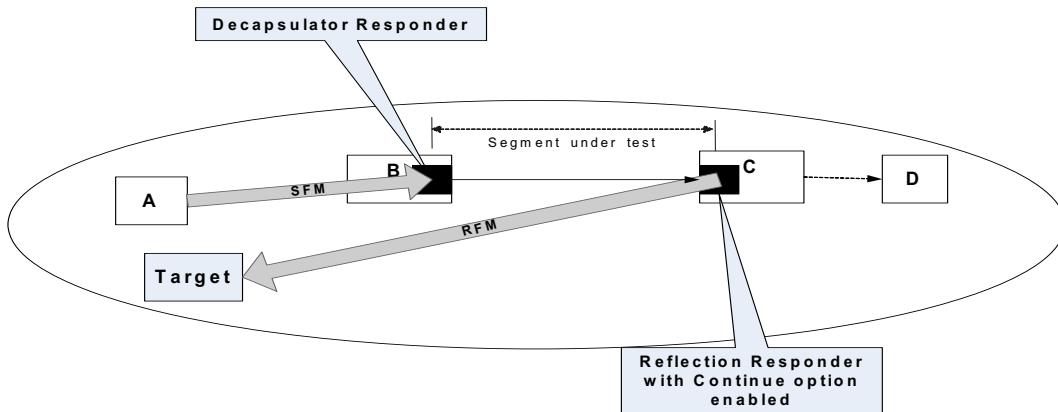


Figure 29-3—Combination of forward path test and return path test

29.2 DDCFM Entity operation

This subclause specifies the following:

- a) Forward path test Reflection Responder
- b) Forward path test RFM Receiver
- c) Return path test Decapsulator Responder
- d) Return path test frames' SFM Originator

29.2.1 DDCFM implementation

The data-driven and data-dependent connectivity fault management (DDCFM) protocol is performed by the Reflection Responder, the Decapsulator Responder, the RFM Receiver, and the SFM Originator. All of these entities are created and activated by operator commands. The operation of Reflection Responder, Decapsulator Responder, and SFM Originator are timer limited.

Both RFMs and SFMs are CFM PDUs and their format is described in 21.4 and 29.4. They are forwarded in the same way as regular CFM messages. An MEP at higher MD level shall drop the received RFMs and SFMs. An MEP at the same level shall process, but not forward, RFMs and SFMs received from its Active SAP, and shall drop RFMs/SFMs received from its Passive SAP. An MFH shall always forward RFMs/SFMs received from its SAPs and shall also process those at the same MD level received from its Active SAP.

The RR, RFM Receiver, DR, and SFM originator are CFM entities that are associated with a specific MD, enabling access only to the administrators of this MD. They can be placed at any point in the network that is bounded by any Domain SAP (DoSAP) of their corresponding MDs. Their relationship to MPs is guided by the MP component entities that the DDCFM entities also require. In particular, the RR does not require any of the MP's sub-functions (19.2 and 19.3) and correspondingly it is defined as an independent CFM shim. It can be placed at any Bridge Port bounded by the DoSAP of the RR's associated MD. It does not even require the EISS Multiplex Entity (6.17) or Backbone Service Instance Multiplexer (6.18). The same holds true for an SFM Originator. Note that both of them transmit CFM frames that are associated with a specific MA but the creation of these DDCFM entities themselves is not associated with an MA. This in practice means that one RR or SFM Originator can send RFMs or SFMs (respectively), which are associated with different MAs than the encapsulated frames. The other two DDCFM entities have a number of common subentities to the MPs. In particular, on top of their unique (MP) RFM Receiver and (MP) Decapsulator Responder state machines, they require the functions provided by the Type Demultiplexer, Level Demultiplexer, and the

OpCode Demultiplexer. In addition if the service is VLAN based, they also require the EISS Multiplex Entity. All of these make the implementation of the RFM receiver or the Decapsulator Responder very simple in the case of an MP where they can reuse the MP component entities and provide the additional functions required by the MP RFM Receiver or MP Decapsulator Responder entities. Implementing these DDCFM entities on non-MPs would require all the previously mentioned MP subentities. Figure 29-5 shows an MP independent RFM Receiver and Figure 29-6 shows an MP independent Decapsulator Responder. On the other hand, implementing these DDCFM entities on MPs would require only the addition of the MP Decapsulator Responder and MP RFM Receiver component entities in the MP architecture with the appropriate changes in the Opcode Demultiplexer as specified in 19.2 and 19.3.

29.2.2 Forward Path Test Reflection Responder

The Reflection Responder (RR) is a CFM protocol shim for filtering frames to be reflected, copying the filtered frames to the Passive SAP if the Continue option is set, and encapsulating each filtered frame with the RFM header (29.4.2). For an interface supporting DDCFM, an RR shim shall be allowed at EISS/ISS SAPs where MPs are allowed on the protocol stack shown in Figure 22-4, Figure 22-8, Figure 26-2, and Figure 26-8. By placing an RR shim on an aggregated IEEE802.3 port within a Bridge Port, the RR can reflect data frames from a member link of a Link Aggregation Group to test if specific data frames can go through the link.

The Figure 29-4 illustrates the component entities of a Reflection Responder.

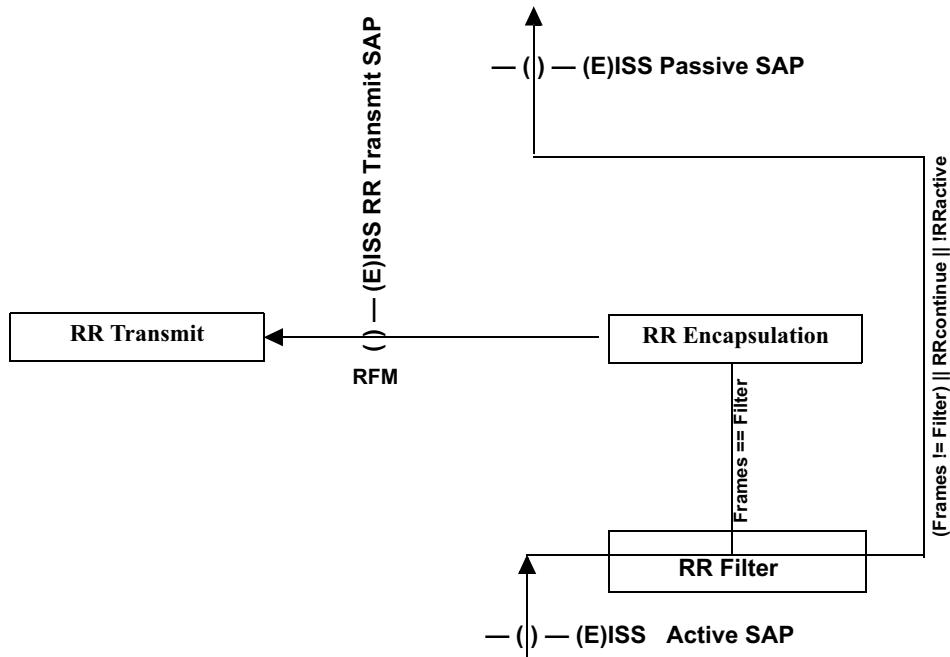


Figure 29-4—Detailed Functions of Reflection Responder

29.2.2.1 RR Filter

The RR Filter function within the Reflection Responder filters frames from Active SAP that match the Reflection Filtering definition(s) [29.2.3 b2)]. When the RR is active and the sampling interval has expired, a frame that matches the Reflection Filter definition(s) is sent to RR Encapsulation to be reflected. A frame is passed to the Passive SAP when RR is not active, the frame does not match the Reflection Filter definition(s), or the Continue option is true.

To prevent data frames from being reflected multiple times within a network, RFM frames cannot be reflected. An RR Filter definition implicitly includes the condition of the OpCode in common CFM header not equal to RFM, in addition to all other conditions specified.

29.2.2.2 RR Encapsulation

The RR Encapsulation function within the Reflection Responder is for encapsulating the filtered frame with the appropriate RFM header. If a sampling interval is specified, once a frame meeting the Reflection Filtering definitions is identified within one sampling interval, no more frames shall be encapsulated until the next sampling interval starts.

29.2.2.3 RR Transmit

RR Transmit identifies the egress port by querying FDB and forwards the RFM frames to the LOM entity of the identified egress port. The query uses the Reflection Target address [12.17.2.1.2, b3] and the vlan_identifier associated with the RFM [29.3.3.3, c]. If there is no vlan_identifier associated with the RFM, the PVID value of the port where RR is configured is used. If an egress port cannot be identified and the FloodingEnabled flag of the RR is true, then the RFM frame is sent to the LOM entity on every port associated with the VID set of the RFM frame. If the FloodingEnabled flag of the RR is false and an egress port cannot be identified, the RFM frame is dropped.

29.2.3 Reflection Responder related parameters

29.2.3.1 Reflection Responder identification

A Reflection Responder is identified by the interface upon which the RR is created, a Maintenance Domain that identifies the administrative boundaries, and a value indicating the direction (Up or Down) in which the RR is facing. The interface upon which the RR is created can be a Bridge Port or a member port of a LAG of a Bridge Port.

29.2.3.2 Maintenance Association for Reflection Responder

The primary VID associated with RR's MA is in the RFM emitted by the RR. When MA for an RR is set to 0, then the VID of the filtered frame are used in the corresponding RFM.

29.2.3.3 Reflection Responder Filter definition

The Reflection Responder Filter is for selecting data frames to be reflected. Multiple filters can be combined together by “&& (and)”, “|| (or)”, or “! (negation)” operations.

All bridges supporting DDCFM shall support the following reflection filters (and may support additional Reflection Filters):

- a) Reflect All—all frames are selected.
- b) VLAN based selection—all data frames associated with the specified VID are selected.
- c) Destination address based selection—all data frames with the specified Destination Address are selected.
- d) Source address based selection—all data frames with the specified Source Address are selected.
- e) EtherType.

29.2.3.4 Sampling interval

The sampling interval controls the maximum rate at which frames can be reflected. If no sampling interval is set, then all frames that arrive while the RR is active and match the filter criteria are reflected. If a sampling

interval is specified then the first matching frame that arrives during the interval is reflected, other matching frames arriving in that interval are not reflected.

29.2.3.5 Continue option

The Continue option [12.17.2.2.2 b5)] allows the selected data frames to be continued to their original destinations by making a copy of the selected frames to be reflected.

The Continue option is set to be true by default, which means that the selected data frames are branched to two directions—one direction to the RR Encapsulation function, and the other direction to the passive SAP. When the Continue option is set to be false, the selected frames shall only be forwarded to the RR Encapsulation function.

29.2.3.6 Duration for Reflection Responder to remain active

Reflecting data frames back into network can create extra traffic in the network. To prevent this action running forever, a duration [12.17.2.1.2 b5)] is used to limit the maximum amount of time for the Reflection Responder to be running. Operator can also specify the maximum number of frames to be reflected. Once activated, RR shall remain active until either its specified duration expires or the maximum number of frames reflected is reached, whichever comes first. An active RR can also be deactivated by operator command before its duration expires.

29.2.3.7 Truncation flag

If the received data frame causes the RFM encapsulated frame to exceed the Maximum Service Data Unit Size (6.5.8), either the data frame is truncated, or the data frame is split into two smaller frames and those two smaller frames are encapsulated by two separate RFM frames. If the Truncation flag is set to be true, the data frame shall be truncated to be encapsulated in one RFM.

29.2.4 Reflection Target and RFM Receiver

The Reflection Target is where the RFM encapsulated reflected frames are forwarded to. The RFM Receiver is a function running on a Reflection Target to pass the received RFMs to their corresponding analyzers.

The Reflection Target has to be within the same Maintenance Domain as the Reflection Responder in order to receive the RFM encapsulated data frames.

When an RFM Receiver is defined on a bridge interface that does not support MP, the RFM Receiver is implemented as a separate CFM entity, as depicted in Figure 29-5 for receiving RFMs from Active SAP. It requires a number of the component entities that are required by an MP. If the RFMs are associated with a VID-based MA, the RFM receiver needs to be placed in a demultiplexer SAP provided by an EISS multiplex entity (6.17).

29.2.5 Return path test related parameters

Return path test configuration has two parts: one part is to configure SFM Originator and the other part is to configure Decapsulator Responder.

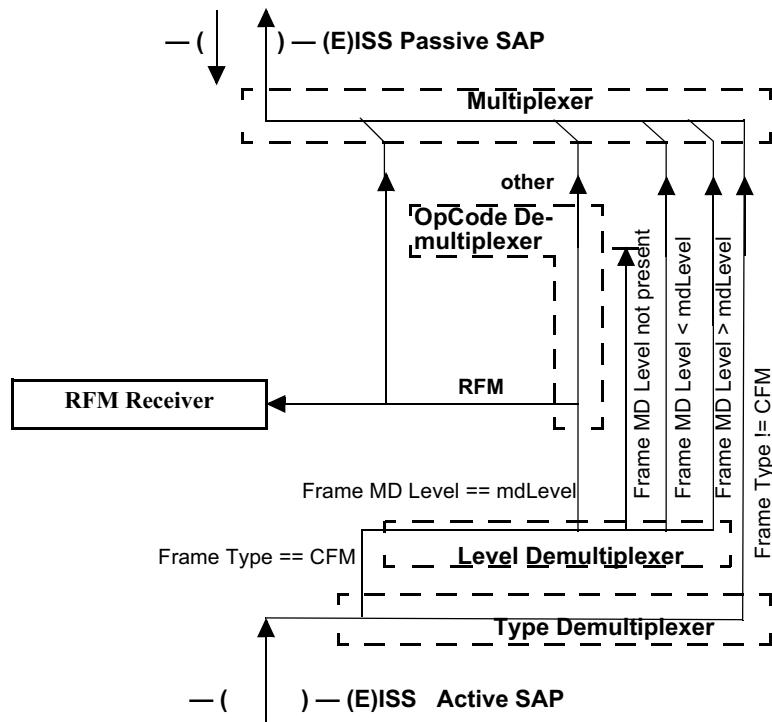


Figure 29-5—RFM Receiver on an non-MP

29.2.6 Decapsulator Responder

The Decapsulator Responder (DR) decapsulates Send Frame Messages (SFMs), and sends the decapsulated frames toward destinations specified by the destination_address field of the decapsulated data frames. Under normal circumstance, the source_address field of the decapsulated frame is replaced by the DR's own MAC address to avoid confusing other bridges' learning in the network. However, a network operator can enforce the DR not to modify the source_address field of the decapsulated frame for special purpose fault diagnosis. The Decapsulator Responder configuration is specified in 12.17.4.1.

DR shim shall be allowed at EISS/ISS SAPs where MPs are allowed on the protocol stack shown in Figure 22-4, Figure 22-8, Figure 26-2, and Figure 26-8.

Multiple DRs can be created on one bridge interface to receive SFMs at different MD levels or SFMs originated from different SFM Originators.

When an SFM is received by an active DR, it is examined for validity and discarded if invalid. If valid, the received SFM shall be decapsulated. The source_address field of the decapsulated frame is replaced by DR's own MAC address unless SourceAddressStayFlag is set. The modified frame shall be used to identify the egress port by querying the FDB. The query uses the destination_address and vlan_identifier values of the decapsulated data frame. If the decapsulated frame is untagged, the PVID value of the port where DR is configured is used. The Boolean FloodingEnabled flag of the DR indicates if the frame is to be flooded or discarded if an egress port cannot be identified by FDB. Figure 29-6 illustrates the detailed functions of Decapsulator Responder. DR shim can be placed at any places where MPs are allowed on the protocol stack as shown in Figure 22-8.

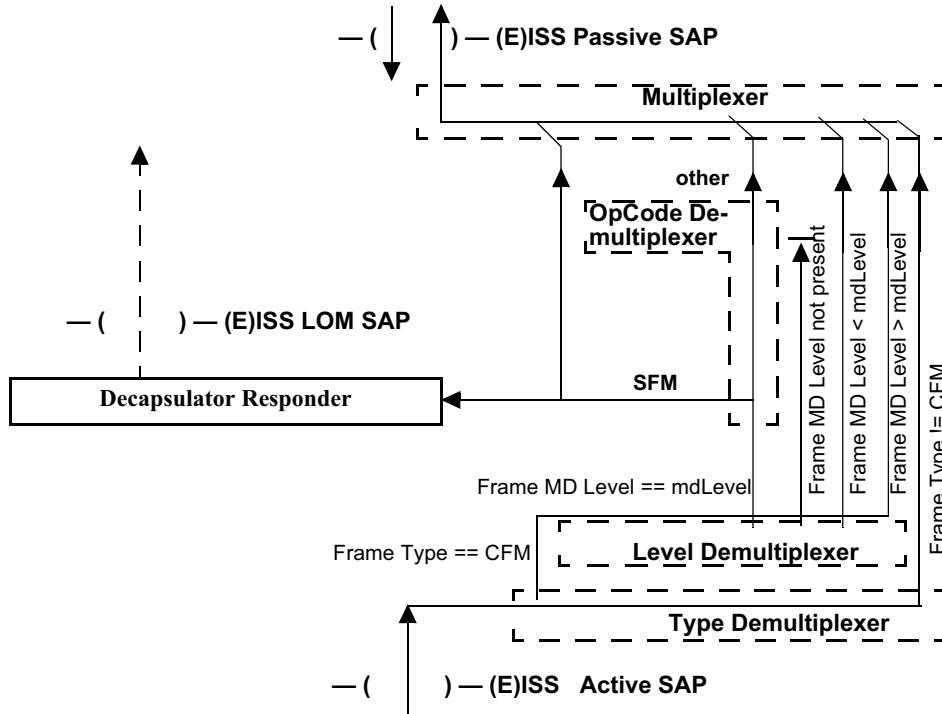


Figure 29-6—Return Path Decapsulator Responder

29.2.7 SFM Originator

SFM Originator is for generating SFMs and sending the SFMs to the corresponding Decapsulator Responder. Each SFM encapsulates one data frame. The SFM Originator configuration is specified in 12.17.5.1.

The bridge interface on which SFM Originator is created does not have to support any MP. However, the SFM Originator has to be configured with the same Maintenance Domain level as the Decapsulator Responder, so that the SFM can be forwarded to the Decapsulator Responder. It is beyond the scope of this standard to address how SFMs are generated.

29.3 DDCFM protocols

This subclause specifies the DDCFM protocols in terms of a number of state machines, the variables and procedures associated with each machine.

29.3.1 Reflection Responder variables

There is one instance of the following parameters per RR, except the nPendingRFMs variable, that is common to all RRs on the bridge components (i.e., per bridge component, but not per RR).

29.3.1.1 RRcontinue

A Boolean variable for the RR's Continue option [12.17.2.2.2, b5] attribute.

29.3.1.2 nPendingRFMs

An integer value used to track the number of RFMs to be transmitted. nPendingRFMs is a variable that can be incremented by every Reflection Responder's RR Encapsulation state machine and can be decremented by RR Transmit state machine.

nPendingRFMs is set to 0 when all of the Reflection Responders on the bridge are deactivated.

29.3.1.3 dataReceived

dataReceived is a Boolean variable, which is set to true when there is a data frame received from (E)ISS, and set to false by the RR Filter state machine.

29.3.1.4 dataFrame

This variable is the data frame received from (E)ISS.

29.3.1.5 nFilteredFrameList

This variable keeps track of the number of filtered data frames to be reflected. This variable is incremented by RR Filter and decremented by Reflection Responder's Encapsulation state machine.

29.3.1.6 filteredFrameList

This is a list of received data frames from RR Filter that have not been processed yet.

29.3.1.7 nextFilteredFrame

This variable copies the first one on the filteredFrameList to be reflected as RFM. Once a frame is copied to this variable, the frame is removed from the filteredFrameList.

29.3.1.8 RRtime

RRtime is the value of RR Duration (29.2.3.6).

29.3.1.9 RRwhile

RRwhile is a timer variable, which is initialized to RR Duration [12.17.2.2.2, b7] when RR is activated and used by the RR Filter state machine to time out the active Reflection Responder. RRwhile has granularity in seconds.

29.3.1.10 RRsampInt

RRsampInt is the sampling interval (29.2.3.4) during which the RR only reflects a maximum of one frame that meets the filter condition. When the sampling interval is not configured, RRsampInt is set to 0.

29.3.1.11 RRsampWhile

A timer variable used by RR Filter state machine to time out sampling interval.

29.3.1.12 RRmaxFrames

The maximum number of frames for RR to reflect after RR is activated [12.17.2.2.2, b8)]. Once activated, RR will stay active until either RRmaxFrames is reached or RRwhile (29.3.1.9) expires, whichever comes first. If RRmaxFrames is set to 0, the RR stays active until RRwhile (29.3.1.9) expires.

29.3.1.13 RRframeCount

A counter used by the Reflection Responder's Encapsulation state machine to keep track of the number of data frames to be reflected. Once RRframeCount is equal to or greater than RRmaxFrames, Reflection Responder is deactivated.

29.3.1.14 filterMatched

A Boolean variable that is set to true if the data frame received matches with the RR's filter conditions and false if the data frame received does not match the RR's filter conditions.

29.3.1.15 RRactive

RRactive is true when the corresponding Reflection Responder is active and false otherwise.

29.3.1.16 nextRFMtransID

The value to place in the RFM Transaction Identifier field of the next RFM to be transmitted. nextRFMtransID is incremented by 1 by the Reflection Responder's processRRencap() procedure with each transmission.

This variable is readable as a managed object [12.17.2.3.3, b.14)].

29.3.1.17 maxDataTLVvalueSize

This is the maximum number of bytes allowed to be copied into Reflected Data TLV's value field in order to keep the RFM's service data unit size not exceeding its Maximum Service Data Unit Size (MSDUS) (6.5.8).

The length of an RFM PDU is 12 octets (4 bytes for CFM header + 1 byte for End TLV(0) + 4 bytes Transaction Identifier + 3 bytes Reflected Data TLV Length/Type field) + the actual number of bytes of the reflected data frame. If there is no other optional TLV for the RFM data frame³⁸, then

- a) maxDataTLVvalueSize = MSDUS – (5 (CFM common header length) + 4 (Transaction Identifier) + 3 (Reflected Data Frame TLV Length/Type field))

29.3.1.18 reflectedDataLength

This variable is to hold the value for the Length field of Reflected Data TLV (29.4.2.4).

29.3.2 RR Filter Procedures

The following procedures are defined for the RR Filter state machine:

- a) filterFrame()
- b) forwardFrame()
- c) loadFilteredFrameList()

³⁸Even though not required, implementers may choose to include other optional TLVs in RFM. If extra TLVs are included in a RFM, the maxDataTLVvalueSize has to be decremented by the number of bytes taken by the extra TLV.

29.3.2.1 forwardFrame()

This procedure passes the frame received to (E)ISS Passive SAP.

29.3.2.2 filterFrame()

filterFrame() is called when RRactive is true and there is a data frame received from EISS. This procedure performs the following steps

- a) If the received data frame from (E)ISS SAP meets the Reflection Filter definition(s), then returns true
- b) Otherwise, returns false

29.3.2.3 loadFilteredFrameList()

This procedure adds the received frame to the filteredFrameList.

29.3.3 RR Encapsulation Procedures

The following procedures are defined for the RR Encapsulation state machine:

- a) processRRencap()
- b) splitFilteredFrame()
- c) constructRFM()

29.3.3.1 processRRencap()

processRRencap() is called when RR is activated and filteredFrameList is not empty. There are two local variables in this procedure: DataFrame_1 and DataFrame_2, which are used when the filtered frame has to be truncated or split into two frames to be reflected.

processRRencap() processes the incoming data frame from the RR Filter as follows:

- a) Set the reflectedDataLength variable to the length of the nextFilteredFrame.
- b) If the value of reflectedDataLength is larger than the bridge's maxDataTLVvalueSize (29.3.1.17), then call splitFilteredFrame (nextFilteredFrame, DataFrame_1, DataFrame_2) and then perform the following two steps:
 - 1) If Truncation flag is set, call constructRFM(TLV_type, maxDataTLVvalueSize, DataFrame_1), where the TLV_type is set to the "Truncated Data TLV" type value defined by Table 21-6
 - 2) If the Truncation flag is not set, perform the following two steps:
 - constructRFM(TLV_type, maxDataTLVvalueSize, DataFrame_1), where the TLV_type is set to the "Data Part 1 TLV" type value defined by Table 21-6
 - constructRFM(TLV_type, reflectedDataLength - maxDataTLVvalueSize, DataFrame_2), where the TLV_type is set to the "Data Part 2 TLV" type value defined by Table 21-6
- c) Otherwise, call constructRFM(TLV_type, reflectedDataLength, nextFilteredFrame), where TLV_type is set to the "Data TLV" type value defined by Table 21-6.

29.3.3.2 splitFilteredFrame()

This procedure is called when the nextFilteredFrame causes the length of RFM's service data unit size to be larger than the Maximum Service Data Unit Size (6.5.8). There are three parameters for this procedure:

- a) nextFilteredFrame, which is an input to the procedure.

- b) DataFrame_1, which contains the first maxDataTLVvalueSize number of bytes from the nextFilteredFrame.
- c) DataFrame_2, which contains the remaining bytes from the nextFilteredFrame.

splitFilteredFrame() performs the following steps:

- Copy the maxDataTLVvalueSize number of bytes from the nextFilteredFrame to DataFrame_1.
- Copy the remaining bytes from the nextFilteredFrame to DataFrame_2.

29.3.3.3 constructRFM()

This procedure is to construct a RFM frame and increment the nPendingRFMs by 1. There are three parameters passed into this procedure:

- a) TLV_Type, which could be 3 (i.e., Data TLV) (21.5.6), 9 (Truncated Data TLV) (21.5.1.1), 10 (Data Part 1 TLV), or 11 (Data Part 2 TLV).
- b) TLV_Length, which is the length of the reflected data frame.
- c) Data frame to be included in the TLV value field.

This procedure constructs the RFM (E)M_UNITDATA.request by the following steps:

- d) Set the source_address parameter to the MP's or bridge interface's own MAC address.
- e) If the Reflection Target Address [12.17.2.2.2, b4] is not specified, then source_address of the selected data frame is copied to the RFM's destination_address parameter; otherwise, Reflection Target Address is copied to the RFM's destination_address parameter.
- f) Set the priority and drop_eligible parameters to the priority and drop_eligible attributes of the Reflection Responder managed object (12.17.2.2.2).
- g) In a VLAN aware bridge, the primary VID associated with the RR's MA [12.17.2.1.2, b1] is used as the vlan_identifier. If the RR's MA is set to 0, the vlan_identifier is determined by the following:
 - 1) If the RR is defined on an S-VLAN component, use the S-VID if the S-VID is present in the selected data frame, otherwise use the PVID of the port where RR is configured.
 - 2) If the RR is defined on a C-VLAN component, use the C-VID if the C-VID is present in the selected data frame, otherwise use the PVID of the port where the RR is configured.
- h) Copy RFM OpCode to the RFM's OpCode Field.
- i) Copy the RR's Maintenance Domain Level [12.17.2.1.2, a] to the RFM's MDLevel field.
- j) Copy the first parameter (TLV_Type) to the Reflected Data TLV Type field.
- k) Copy the second parameter (TLV_Length) to the Reflected Data TLV Length field.
- l) Copy the third parameter (Data frame) to the Reflected Data TLV value field.
- m) Copy the nextRFMtransID to the RFM Transaction Identifier field. Increment nextRFMtransID by 1, wrapping around from $2^{32} - 1$ to 0.
- n) Increment nPendingRFMs by 1.
- o) Set the frame_check_sequence parameter to the unspecified value; sets the connection_identifier to null.

29.3.4 RR Transmit procedure

The following procedure is defined for the RR Transmit state machine:

- a) xmitRFM()

29.3.4.1 xmitRFM()

xmitRFM() is called by the RR Transmit state machine. It queries the FDB in order to identify an egress port. The set of potential transmission ports, normally created by the Active Topology enforcement (8.6.1) in

IEEE Std 802.1Q, is the set of all bridge interfaces that are both in the active set of the `vlan_identifier` associated with the RFM, and that are in the Forwarding state for that `vlan_identifier`.

- a) Query FDB using the RFM's `destination_address` field and `vlan_identifier` parameter values.
- b) If an egress port is identified and the identified Egress port meets the following conditions:
 - 1) The Egress port does not have a Down MEP with higher MD level than the RFM's MD;
 - 2) The Egress port does not have an Up MEP with same or higher MD level as the received RFM's MD;
 then send an RFM `EM_UNITDATA.request` to the egress port's LOM SAP. Otherwise, drop the received RFM frame;
- c) If an egress port cannot be identified
 - 1) When the `FloodingEnabled` flag of the RR [12.17.2.1.2, b7] is not set, drop the frame.
 - 2) When the `FloodingEnabled` flag of the RR is set, send `M_UNITDATA.request` to the LOM's SAP of every forwarding port that meets the conditions 1) and 2) of item b) above.

29.3.5 Reflection Responder related state machines

29.3.5.1 RR Filter state machine

Figure 29-7 depicts the state machine for RR Filter.

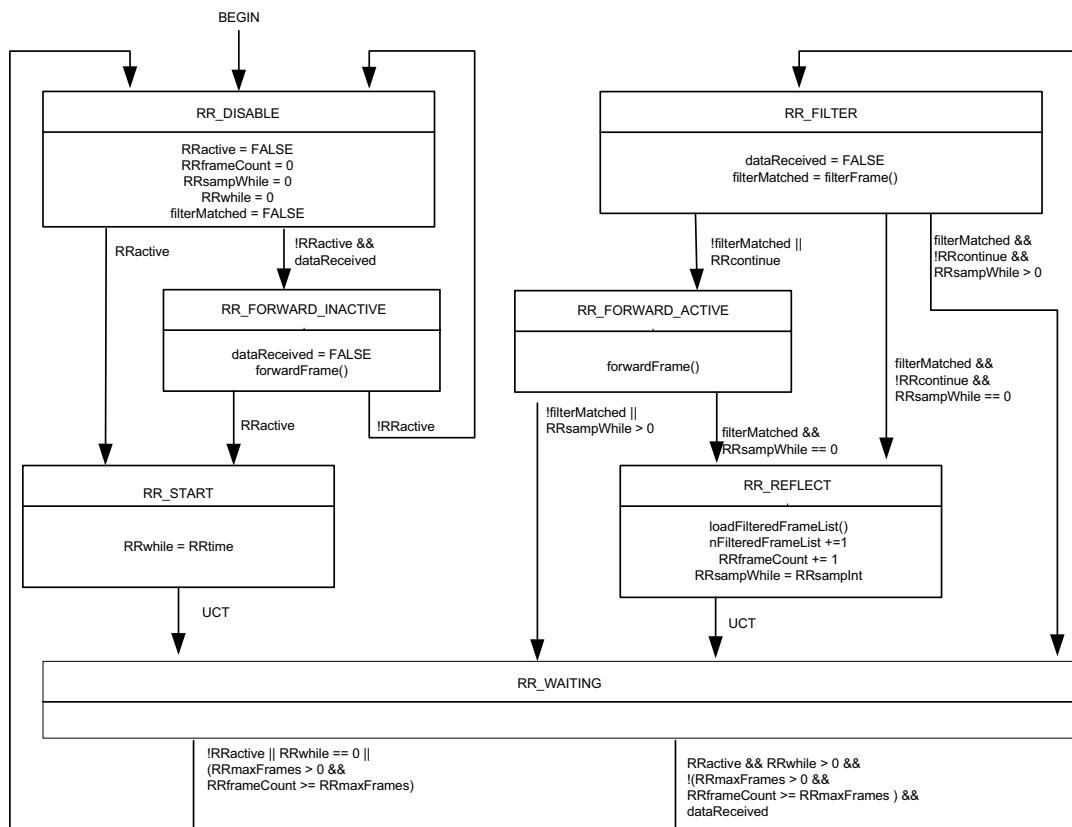


Figure 29-7—RR Filter state machine

29.3.5.2 RR Encapsulation state machine

Figure 29-8 describes the state machine for RR Encapsulation.

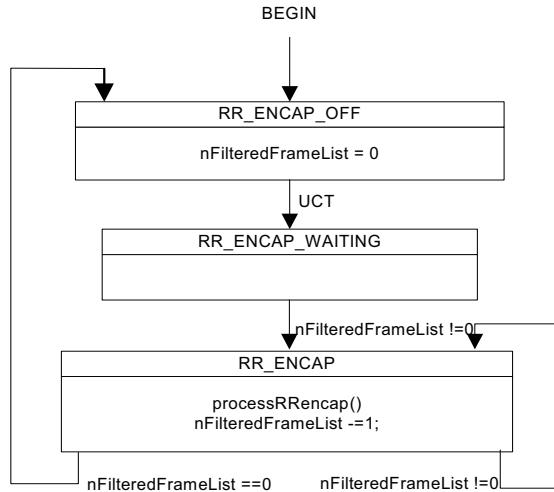


Figure 29-8—RR Encapsulation state machine

29.3.5.3 RR Transmit state machine

RR Transmit state machine can be shared by multiple Reflection Responders activated on the bridge.

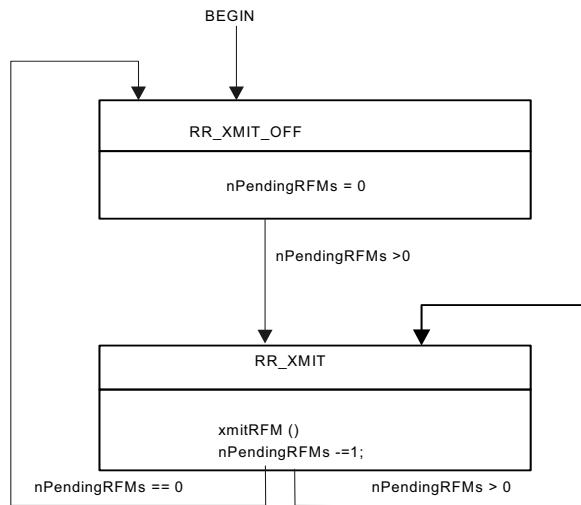


Figure 29-9—RR Transmit state machine

29.3.6 RFM Receiver variables

The following variables are local to the RFM Receiver state machine. There is one instance of these variables per RFM Receiver.

29.3.6.1 RFMreceived

RFMreceived is a Boolean variable, which is set to true by the MP OpCode Demultiplexer when an RFM is received, and cleared by the RFM Receiver state machine.

29.3.6.2 RFMPDU

RFMPDU is a record of the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the last RFM received by the MP OpCode Demultiplexer (19.2.7).

29.3.7 RFM Receiver procedure

The procedure listed in 29.3.7.1 and 29.3.7.2 is defined for the RFM receiver.

29.3.7.1 ReceiveRFM()

ReceiveRFM() procedure performs the following steps:

- a) If the destination_address field of the received RFM does not match the RFM Receiver's MAC address, drops the frame.
- b) Otherwise, pass the received RFMPDU to RFM Analyzer.³⁹

Even though RFM Analyzer is out of the scope of this standard, RFM Analyzer has to be able to recognize the value in Reflected Data TLV (29.3.3.2), which could be

- A whole data frame being reflected; or
- A truncated data frame being reflected; or
- The first portion of the data frame being reflected; or
- The second portion of the data frame being reflected.

If an RFM Receiver is configured on an MP and the RFM Receiver does not receive any RFMs within the expected time frame, it can send a LBM (21.7) to the MP on which the Reflection Responder is defined to verify the connectivity to the MP. If there is no LBR received, ReceiveRFM() can report an alarm to the network administrator.

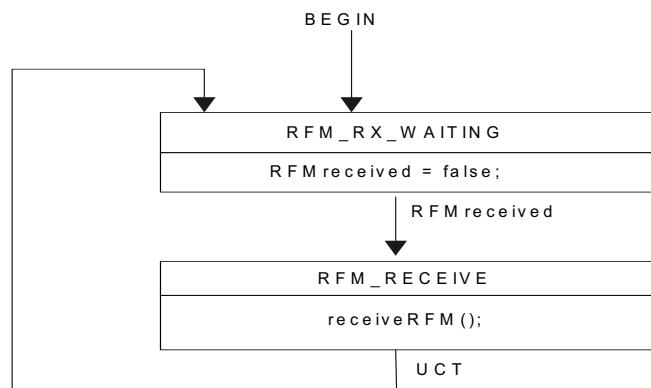
29.3.7.2 RFM Receiver state machine

Figure 29-10—RFM Receiver state machine

³⁹RFM Analyzer is out of the scope of this standard.

29.3.8 Decapsulator Responder variables

There is one instance of the parameters listed in 29.3.8.1 through 29.3.8.7 per Decapsulator Responder.

29.3.8.1 DRwhile

A timer variable used by the Decapsulator Responder state machine to time out the active Decapsulator Responder. DRwhile has a granularity finer than or equal to 1 s.

29.3.8.2 DRtime

The time that the Decapsulator Responder can remain active after its activation.

29.3.8.3 SFMPDU

SFMPDU is a record of the M_UNITDATA.indication or EM_UNITDATA.indication parameters of the SFM received by the MP OpCode Demultiplexer (19.2.7) when an SFM is received.

29.3.8.4 DRactive

DRactive variable represents the activation status of the corresponding Decapsulator Responder. DRactive is set to true when the corresponding Decapsulator Responder is activated and set to false when the corresponding Decapsulator Responder is deactivated or its duration expires. When DRactive is false, all SFMs received by the bridge interface are dropped.

29.3.8.5 SFMreceived

SFMreceived is set to true by the OpcodeDemultiplexer (19.2.7) when an SFM is received.

29.3.8.6 previousSFMtransId

previousSFMtransId variable keeps the value of SFM Transaction Identifier of the previously received SFM frame. The previousSFMtransId variable is initialized to zero when the DR is activated.

29.3.8.7 SFMsequenceErrors

SFMsequenceErrors variable keeps the total number of out-of-sequence SFMs received. SFMsequenceErrors is initialized to zero when the DR is activated.

29.3.9 Decapsulator Responder procedures

The Decapsulator Responder has the following procedures:

- a) ProcessSFM()
- b) DropSFM()

29.3.9.1 processSFM()

Called by the Decapsulator Responder state machine when an SFM is received and the Decapsulator Responder is active. processSFM() processes the incoming SFM in SFMPDU as follows:

- a) The SFM is examined for validity using the methods used for CFM as described in (20.46) and discarded if invalid.

- b) If the destination_address of the SFM does not match the DR's own MAC address or the source_address of the SFM does not match with DR's SFM Originator [12.17.4.3.2 b2)], the frame is discarded.
- c) If the SFM frame passes these tests, the frame carried in its Original Data TLV (29.4.3.4) is decapsulated to create an (E)M_UNITDATA.request as follows:
 - 1) The destination_address parameter is set to the destination_address of the frame in the Original Data TLV.
 - 2) If the SourceAddressStayFlag [12.17.4.3.2 b1)] is false, the source_address parameter is set to the DR's MAC address, otherwise the source_address parameter is set to the source_address of the frame in the Original Data TLV.
 - 3) The vlan_identifier, priority, drop_eligible or canonical_format_indicator parameters and mac_service_data_unit parameter are set as follows:
 - If the DR is configured on a port of an S-VLAN component, and the TPID of the frame in the Original Data TLV indicates the Service VLAN tag, the vlan_identifier, priority, and drop_eligible parameters are taken from the corresponding values in the TCI and the mac_service_data_unit is taken from the remainder of the frame following the TCI and excluding the last four octets (FCS).
 - If the DR is configured on a port of a C-VLAN component, and the TPID in the Original Data TLV indicates a Customer VLAN tag, the vlan_identifier and priority parameters are taken from the corresponding values in the TCI, the canonical_format_indicator parameter is set to 0, and the mac_service_data_unit is taken from the remainder of the frame in the Original Data TLV following the TCI and excluding the last four octets (FCS).
 - Otherwise the vlan_identifier is set to the PVID for the port on which the DR is configured, the priority is set to the priority of the SFM, the drop_eligible parameter is set to the value of drop_eligible for the SFM, the canonical_format_indicator is set to 0, and and mac_service_data_unit is taken from the frame in the Original Data TLV immediately following the source_address and excluding the last four octets (FCS).
 - 4) Sets the frame_check_sequence parameter to the unspecified value (6.6.1).
 - 5) Sets the connection_identifier parameter to null.
- d) ProcessSFM() queries the Filtering Database using the destination_address and vlan_identifier parameters of the (E)M_UNITDATA.request. The set of potential transmission ports, normally created by the Active Topology enforcement (8.6.1), is the set of all bridge ports that are both in the member set of the vlan_identifier and that are in the Forwarding state for that vlan_identifier.
 - 1) If an Egress Port is identified, the decapsulated frame is carried to the related LOM SAP for transmission;
 - 2) If an Egress Port is not identified and the FloodingEnabled flag of the DR is set, the decapsulated frame is flooded to the LOM SAP of all forwarding ports in the VID set; otherwise the frame is discarded;
- e) If the received SFM is not the first frame received by the DR and the value of SFM Transaction Identifier field in the received SFM is not 1 greater than the value saved in the previousSFMtransId, increment SFMsequenceErrors by 1.
- f) Set the value of previousSFMtransId to the value of the SFM Transaction Identifier field of the received SFM.

29.3.9.2 DropSFM()

Called by Decapsulator Responder state machine whenever the Decapsulator Responder is not active in order to silently discard the received SFM frames.

29.3.10 Decapsulator Responder state machine

Once the Decapsulator Responder is configured and activated by network administrator, it remains active until the configured timer expires or being deactivated by the network administrator (see Figure 29-11).

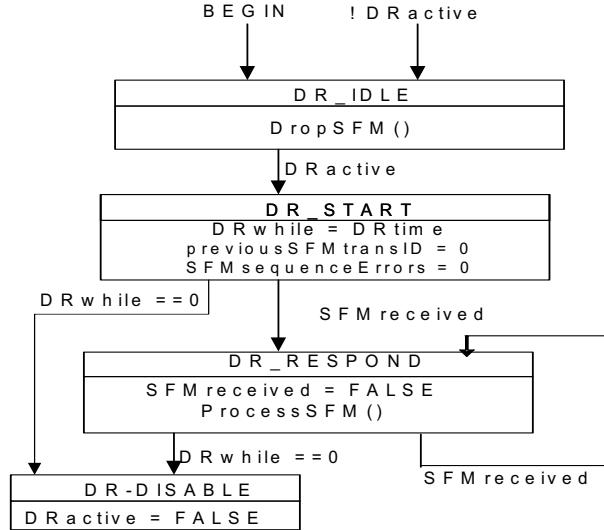


Figure 29-11—Decapsulator Responder state machine

29.4 Encoding of DDCFM Protocol Data Units

This subclause specifies the method of encoding DDCFM PDUs. The specifications include the following:

- The format of RFM
- The format of SFM

29.4.1 RFM and SFM Header

The RFM Header and SFM Header are same as the Common CFM Header defined by Table 21.3, with extra OpCodes defined for RFM and SFM (Table 21-4).

29.4.2 RFM format

The RFM format is shown in Table 29-1.

29.4.2.1 Flags

(1 octet) in RFM. The Flags field of the Common CFM Header for RFM is reserved for future use.

29.4.2.2 First TLV Offset

(1 octet) The First TLV Offset field of the Common CFM Header in a RFM is transmitted as 4.

Validation Test: The First TLV Offset field of the Common CFM Header in a RFM contains a value greater than or equal to 4.

Table 29-1—RFM format

Field	Octet
Common CFM Header	1–4
RFM Transaction Identifier	5–8
Reserved for definition in future versions of the protocol ^a	
Reflected Data TLV	First TLV Offset + 5
Additional TLV	First TLV Offset + 5 + Length of Reflected Data TLV
End TLV (0)	

^aThis field has 0 length in this version 0 of DDCFM. It is shown in order to stress that additional information can be present in future versions of DDCFM, and that a version 0 receiver ignores its contents, if present.

29.4.2.3 RFM Transaction Identifier

(4 octets) A Reflection Responder copies the content of nextRFMtransID variable (29.3.1.16) to this field.

29.4.2.4 Reflected Data TLV

Reflected Data TLV in RFM is a Data TLV (21.5.6) to contain the reflected data frame. The Length field is the total octets of the reflected data frame.

The Type field of the Reflected Data TLV could be one of the following four values (Table 21-6):

- a) Data TLV Type value, which is 3
- b) Truncated Data TLV Type value, which is 9
- c) Data-Part-1 TLV Type value, which is 10
- d) Data-Part-2 TLV Type value, which is 11

29.4.2.5 Additional RFM TLVs

The following TLVs may be included in a RFM by each MP originating RFM (Reflection Responder):

- a) Send ID TLV (21.5.3) and
- b) Organization-Specific TLVs (21.5.2)

The Additional RFM TLVs may be placed in any order.

29.4.3 SFM format

The SFM format is shown in Table 29-2.

29.4.3.1 Flags

(1 octet) in SFM. The Flags field of the Common CFM Header for SFM is reserved for future use.

Table 29-2—SFM format

Field	Octet
Common CFM Header	1–4
SFM Transaction Identifier	5–8
Reserved for definition in future versions of the protocol ^a	
Original Data TLV	First TLV Offset + 5
Additional TLVs	First TLV Offset + 5 + Length of Original Data TLV
End TLV (0)	

^aThis field has 0 length in this version 0 of DDCFM.

29.4.3.2 First TLV Offset

(1 octet) The First TLV Offset field of the Common CFM Header in an SFM is transmitted as 4.

Validation Test: The First TLV Offset field of the Common CFM Header in an SFM contains a value greater than or equal to 4.

29.4.3.3 SFM Transaction Identifier

(4 octets) SFM originator may use this field to distinguish the order of SFMs sent. DR may use this identifier to check the order of the received SFMs.

29.4.3.4 SFM Original Data TLV

SFM Original Data TLV is a Data TLV (21.5.6) to contain the original data frame. The Length field is the total octets of the original data frame, including its destination_address, source_address, TPID, TCI, Length/Type, client data, pad ,and FCS fields.

29.4.3.5 Additional SFM TLVs

The following TLVs may be included in an SFM by each SFM Originator:

- a) Send ID TLV (21.5.3)
- b) Organization-Specific TLVs (21.5.2)

The Additional SFM TLVs may be placed in any order.

30. Principles of congestion notification

Congestion notification comprises capabilities for detecting and mitigating queue congestion for selected classes of traffic in Virtual Bridged Local Area Networks. These capabilities can be used in networks with a bandwidth-delay product on the order of 5 Mbits or less in order to decrease the likelihood of frame loss for Congestion Controlled Flows (CCFs). As the geographical size, per-hop delay, and/or maximum hop count of a network grow, causing the bandwidth-delay product of the network to increase beyond this value, oscillations in buffer occupancy begin, and their amplitude increases gracefully with bandwidth-delay product. If the parameters controlling the algorithm are not adjusted accordingly and the sizes of the bridges' buffers are not increased, frames are then likely to be lost due to congestion.

Congestion notification depends on the formation of a cooperating set of systems including VLAN-aware Bridges and end stations to achieve the reduction in frame loss. By partitioning the bridges' and end stations' resources, frames sourced or sunk by noncooperating end stations or bridges can be carried with minimal contention with CCFs for those resources. Accordingly, congestion notification entities (Clause 31) are specified as shims that make use of and provide the ISS or EISS (6.6, 6.8) at Service Access Points (SAPs) within the network. The operation of the congestion notification protocol is described in Clause 32, and the formats of the Congestion Notification Message and Congestion Notification TLV are described in Clause 33.

This clause provides the context necessary to understand the congestion notification protocol: how congestion controlled regions of networks are identified; how CCFs are identified and separated from other frames; and how congestion notification entities in bridges and end stations operate.

This clause introduces the concepts essential to congestion notification as follows:

- a) The problem and requirements on the solution are presented (30.1).
- b) The basic principles of the Quantized Congestion Notification protocol (QCN) are presented, including descriptions of a CP that sends a Congestion Notification Message (CNM) to a Reaction Point (RP) (30.2).
- c) A CCF consists of frames, all with the same priority value, and all assigned to a single Flow queue and RP in the originating end station (30.3).
- d) A Congestion Notification Priority Value (CNPV) is one of the eight priority values that has been configured exclusively for the use of CCFs in the congestion-aware systems of a Virtual Bridged LAN (30.4).
- e) A CN-TAG can be transmitted with every data frame in a CCF, and is returned in every CNM (30.5).
- f) A CND is constructed as a subset of bridges and end stations in a Virtual Bridged LAN, all of which are configured to handle a particular CNPV, the resources of which subset are defended by the bridges whenever the offered load exceeds the network capacity (30.6).
- g) The relationship of Multicast data to congestion notification is explained (30.7).
- h) The peering relationship of RPs and CPs in Provider Bridged and Provider Backbone Bridged Networks is explained (30.8).

30.1 Congestion notification design requirements

The congestion notification protocol, congestion notification algorithm (QCN), and supporting protocols specified in this standard meet requirements for the following:

- CCF and network performance.
- Limiting the implementation complexity, of both RPs and CPs.
- Not over-constraining the design of Bridged Local Area Networks using CCF.

- Facilitating interoperability, coexistence, and deployment into existing networks, and limiting the administrative effort associated with using CCF in a network.

NOTE 1—QCN is designed to operate over a wide range of conditions. This subclause (30.2) necessarily uses qualitative terms such as “low probability,” “small,” “stable,” and “modest” (for brevity). The bibliography (Annex M) provides references (Alizadeh, et al. [B1]) to studies of QCN performance for various network and traffic configurations, using both simulation and a fluid model (see 30.3 for the necessary equations, and for network sizing recommendations).

The following performance requirements are met for s CCFs (30.3):

- a) There is a very low probability of frame discard due to buffer overflow at a CP.
- b) The average buffer occupancy at each CP that is a Current Congestion Point (CCP) for one or more CCFs is a small fraction of the bandwidth delay product, and largely independent of the number of CCFs, thus ensuring that loss avoidance does not result in large and fluctuating transit delays.
- c) The buffer occupancy at a CCP has a low probability of underflow, i.e., of becoming empty and thus failing to use available bandwidth when an RP is rate limiting a CCF as a result of CNMs received from that CCP.
- d) When multiple CCFs with a single CCP share that CCP, they obtain very nearly the same share of the bandwidth. When multiple CCFs have one or more CCPs in common they each obtain a share of bandwidth resources proportional to their share of contested resources.
- e) The low probability of buffer overflow and underflow at a CCP is maintained for both small and large numbers of CCFs, and when CCFs or the load offered by a given CCF changes.
- f) The bandwidth provided to each CP by its attached LAN will be used, at all times, i.e., if the CP's buffer is not empty it will transmit, given the available bandwidth. The fact that the bandwidth available to one CCF passing through a given CP can be constrained by another CCP will not reduce the bandwidth through that CP for other CCFs not sharing the CCP.

NOTE 2—The probability of frame loss in the absence of CCF naturally depends on the response of higher layer protocols to increasing transit delay, and the importance of avoiding loss on their response. Highly loss sensitive protocols typically lack loss avoidance mechanisms, and no such mechanisms are assumed in the analysis of QCN.

NOTE 3—Requirements a) through d) can be summarized by characterizing the performance of QCN as stable, responsive, and (proportionally) fair with respect to buffer occupancy processes. Requirements e) and f) demand that available resources are used, and that stability not depend on having a large number of slowly changing CCFs.

The following requirements limit the complexity and hence the cost of Bridge Port (CP) support for CCF:

- g) The CCF state retained by each Bridge Port is fixed, and determined solely by the number of congestion management transmission queues (i.e., by the number of CPs, one per CNPV, up to a maximum of 7) and the per CP requirements of the congestion notification algorithm (QCN).
- h) A CP does not retain any per CCF or per RP state, nor is it required to allocate any data frame passing through the CP to a CCF or an RP. All the information necessary for a CP to address a CNM to an end station, and for that end station to identify the CCF and RP from the CNM, is contained in each frame of a CCF.
- i) The buffering required, at each CP, to avoid frame loss should be a small fraction of the network's bandwidth delay product.

The following requirements limit the complexity and hence the cost of end station (RP) support for CCF:

- j) The CCF state retained by each end station for each congestion managed transmission queue, i.e., for each RP, is fixed.
- k) The end station state requirements of QCN are determined solely by the number of RPs, and are independent of the number of CCFs and the number of CPs in the network.

- l) The number of CCFs, the identification of each CCF, the allocation of transmitted frames to CCFs, and the allocation of CCFs to RPs is determined by each end station independently.
- m) The end station is not required to transmit any additional frames, as a consequence either of transmitting or of receiving a frame for a CCF.
- n) The CN-TAG for each CCF is independent of changes in RP state information.
- o) The CCF protocol and QCN do not use any higher-layer protocol information carried in frames.
- p) QCN calculations are simple and can be performed rapidly: i.e., the number of operations required to process each event is fixed, and all multiplications and divisions can be reduced to the use of factors of 2 or the use of a fixed and small lookup table. An RP does not attempt to calculate control loop gains and delays and other, potentially rapidly changing CCF dependent variables.

NOTE 4—A CP or an RP can retain additional state for the purposes of management, but this state is not required by the CCF protocol or QCN.

NOTE 5—An end station can use higher layer protocol information to decide whether a frame is to be allocated to a CCF, and to select a CCF.

The following requirements ensure that CCF does not over-constrain the design of the network, or limit the use of protocols that determine the active topology of the network:

- q) RPs and CPs are unaware of the path taken by frames through the network, require no signalling from the network of changes in path, and take no exceptional action when paths change.
- r) An RP does not assume that all frames that it transmits, or any identifiable subset of those frames such as those identified by the end station as belonging to a single CCF or transmitted to the same destination, are constrained to follow the same path through the network. An RP can transmit frames to many different destinations, and transmission to a given destination can be multi-path.

The following requirements support interoperability, facilitate deployment, and limit the administrative effort required to use CCF:

- s) The boundaries of each CND are determined automatically.
- t) The destination of a CCF can lie outside the transmitting end station's CND, and the destination end station can be unaware of the use of CCF within the CND. Incremental deployment of CCF is therefore possible.
- u) The feedback provided in each CNM provides the receiving RP with the information needed to act on that CNM, and does not require the RP to know or interpret information about the sending CP.
- v) Provider Backbone Bridged Networks can be interposed into a CND, in order to reduce the number of MAC addresses learned by bridges in environments with large numbers of (perhaps virtual) end stations.

These requirements have a number of additional consequences for the design of QCN, including the following:

- w) Additional mechanisms to determine round trip time are not required, as the algorithm is robust over its targeted range of round trip time.
- x) QCN cannot regulate frame transmission by acknowledgements, as does TCP/IP.

30.2 Quantized Congestion Notification protocol

This subclause provides an overview of the baseline simulation of the QCN algorithm (Alizadeh, et al.) [B1] used to develop this standard. This introduction provides no normative text. It will focus on the key features of the QCN algorithm, omitting the normative details given in the rest of this standard.

The QCN algorithm is composed of the following two parts:

- Congestion Point (CP) Algorithm:** this is the mechanism by which a congested bridge or end station buffer samples outgoing frames and generates a feedback message (Congestion Notification Message or CNM, 33.3, in this standard) addressed to the source of the sampled frame. The feedback message contains information about the extent of congestion at the CP.
- Reaction Point (RP) Algorithm:** this is the mechanism by which a Rate Limiter (RL) associated with a source decreases its sending rate based on feedback received from the CP, and increases its rate *unilaterally* (without further feedback) to recover lost bandwidth and probe for extra available bandwidth.

30.2.1 The CP Algorithm

A bridge containing a CP is modeled as an ideal output-queued bridge. The CP buffer is shown in Figure 30-1. The goal of the CP is to maintain the buffer occupancy at a desired operating point, Q_{eq} .⁴⁰ The CP computes a congestion measure F_b (defined below) and, with a probability depending on the severity of congestion, selects a frame from the incoming stream and sends the value of F_b in a feedback message to the source of the selected frame.

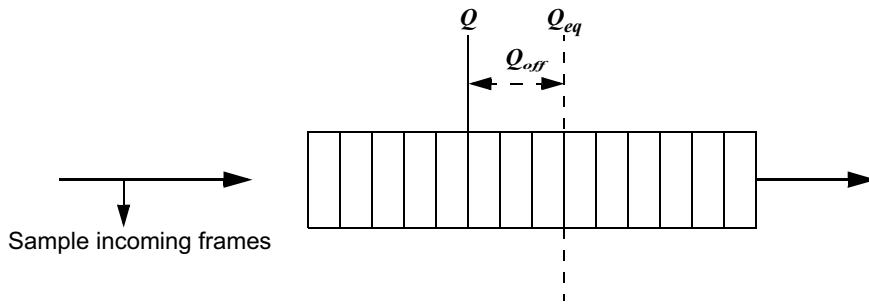


Figure 30-1—Congestion detection in QCN CP

Let Q denote the instantaneous queue size and Q_{old} denote the queue size when the last feedback message was generated. Let $Q_{off} = Q - Q_{eq}$ and $Q_\delta = Q - Q_{old}$.

Then F_b is given by Equation (1).

$$F_b = -(Q_{off} + wQ_\delta) \quad (1)$$

where w is a non-negative constant, taken to be 2 for the baseline simulation. The value of F_b is quantized to 6 bits before transmission, based on Q_{eq} and w .

The interpretation is that F_b captures a combination of queue size excess (Q_{off}) and rate excess (Q_δ). Indeed, $Q_\delta = Q - Q_{old}$ is the *derivative* of the queue size and equals input rate less output rate. Thus, when F_b is negative, the buffer is oversubscribed. When $F_b < 0$, Figure 30-2 shows the probability with which a congestion message is reflected back to the source as a function of $|F_b|$. The feedback message contains the value of F_b , quantized to 6 bits. When $F_b \geq 0$, there is no congestion and no feedback messages are sent.

⁴⁰In simulation, setting Q_{eq} to 20% of the physical buffer size (150 000 octets) produced good results.

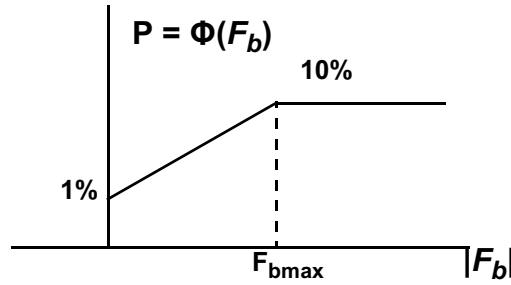


Figure 30-2—Sampling (reflection) probability in QCN CP as a function of $|F_b|$

30.2.2 Basic Reaction Point algorithm

The RP is not given positive rate-increase signals by the network. Also, there are no acks at Layer 2 to trigger rate increases. Therefore, the RP requires a local mechanism to determine when, and by how much, the send rate is increased. Before proceeding to explain the RP algorithm, we will need the following terminology:

- **Current Rate (CR):** The transmission rate of the Rate Limiter (RL) at any time.
- **Target Rate (TR):** The sending rate of the RL *just before* the arrival of the last feedback message, and the new goal for the Current Rate.
- **Byte Counter:** A counter at the RP for counting the number of bytes transmitted by the RL. It triggers rate increases by the RL. See 30.2.2.2.
- **Timer:** A clock at the RP that is also used for triggering rate increases at the RL. The main purpose of the timer is to allow the RL to rapidly increase the rate when its sending rate is very low and bandwidth becomes available. See 30.2.3.

We now explain the RP algorithm assuming that only the byte counter is available. Later, we will briefly explain how the timer is integrated into the RP algorithm. Figure 30-3 shows the basic RP behavior.

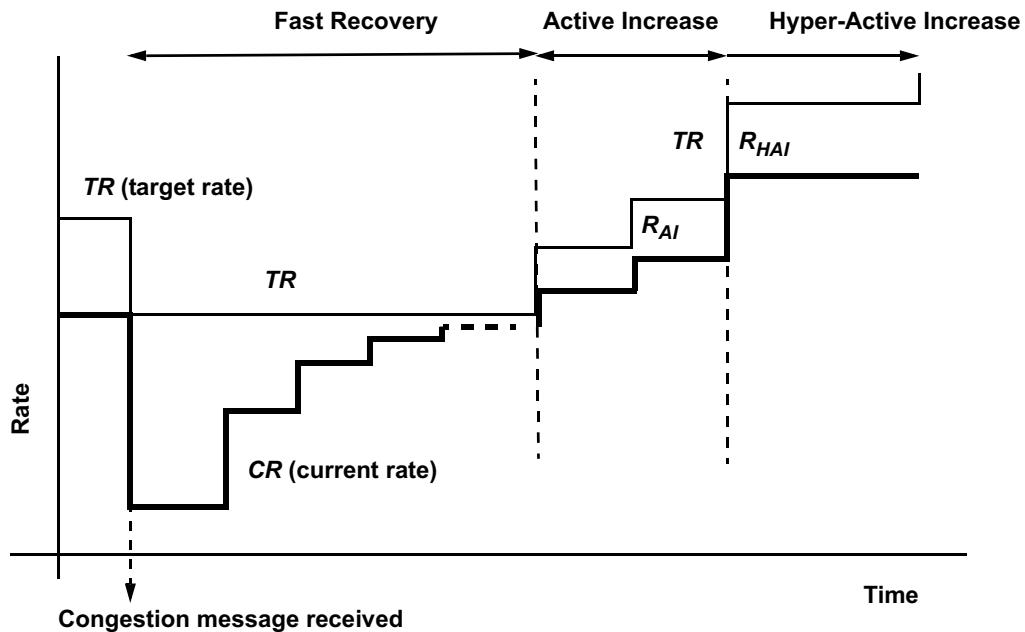


Figure 30-3—QCN RP operation

30.2.2.1 Rate decreases

Rate decreases occur only when a feedback message is received, in which case CR and TR are updated as follows in Equation (2) and Equation (3):

$$TR \leftarrow CR \quad (2)$$

$$CR \leftarrow CR(1 - G_d|F_b|) \quad (3)$$

where the constant G_d is chosen so that $G_d|F_{b\max}| = \frac{1}{2}$, i.e., the sending rate can decrease by at most 50%.

30.2.2.2 Rate increases

Rate increases occur in two phases: Fast Recovery and Active Increase.

Fast Recovery (FR). The byte counter is reset every time a rate decrease is applied and the RP enters the FR state. FR consists of 5 cycles, each cycle equal to 150 KBytes of data transmission by the RL. The RP counts the cycles of the byte counter. At the end of each cycle, TR remains unchanged while CR is updated as follows in Equation (4):

$$CR \leftarrow \frac{1}{2}(CR + TR) \quad (4)$$

The cycle duration of 150 KBytes is chosen to correspond to the transmission of 100 frames, each 1500 Bytes long. The idea is that when the RL has transmitted 100 frames and, given that the minimum sampling probability at the CP is 1%, if it has not received a feedback message then it may infer that the CP is uncongested. Therefore it increases its rate as above, recovering some of the bandwidth it lost at the previous rate decrease episode. Thus, the goal of the RP in FR is to *rapidly recover* the rate it lost at the last rate decrease episode.⁴¹

Active Increase (AI). After 5 cycles of FR have completed, the RP enters the Active Increase (AI) phase where it probes for extra bandwidth on the path. During AI, the byte counter counts out cycles of 50 frames (this can be set to 100 frames for a less frequent probing). At the end of each cycle, the RL updates TR and CR as follows in Equation (5) and Equation (6):

$$TR \leftarrow TR + R_{AI} \quad (5)$$

$$CR \leftarrow \frac{1}{2}(CR + TR) \quad (6)$$

where R_{AI} is a constant chosen to be 5 Mbps in the baseline simulation.

30.2.3 RP algorithm with timer

Since rate increases using the Byte Counter occur at a frequency proportional to the current sending rate of the RL, when CR is very small, the duration of Byte Counter cycles when measured in seconds can become unacceptably large. Since the speed of bandwidth recovery (or responsiveness) is a key performance metric, a Timer was included in QCN.

The Timer functions similarly as the Byte Counter: it is reset when a feedback message arrives, enters FR and counts out 5 cycles of T ms duration (T is 10 ms long in the baseline), and then enter AI where each cycle is T/2 ms long.

The Byte Counter and Timer jointly determine rate increases at the RL as shown in Figure 30-4. After a feedback message is received, they each operate independently and execute their respective cycles of FR and AI. Together, they determine the state of the RL and its rate changes as follows:

⁴¹The BIC-TCP algorithm [B3] has a similar Fast Recovery mechanism.

- a) The RL is in FR if *both* the Byte Counter and the Timer are in FR. In this case, when either the Byte Counter or the Timer completes a cycle, CR is updated according to Equation (4).
- b) The RL is in AI if *only one* of the Byte Counter and the Timer is in AI. In this case, when either completes a cycle, TR and CR are updated according Equation (5) and Equation (6).
- c) The RL is in HAI (for Hyper-Active Increase) if *both* the Byte Counter and the timer are in AI. In this case, the *i*th time that a cycle for either the Byte Counter or the Timer is completed, TR and CR are updated as shown in Equation (7) and Equation (8):

$$TR \leftarrow TR + iR_{HAI} \quad (7)$$

$$CR \leftarrow \frac{1}{2}(CR + TR) \quad (8)$$

where R_{HAI} is constant set to 50 Mbps in the baseline simulation (Figure 30-3). So the increments to TR in HAI occur in multiples of 50 Mbps.

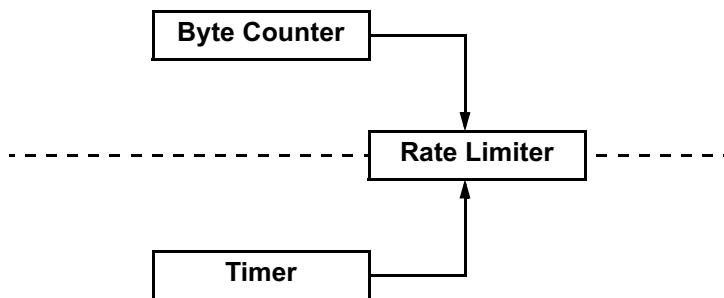


Figure 30-4—Byte Counter and Timer interaction with Rate Limiter

Thus, the Byte Counter and Timer should be viewed as providing the RL opportunities to increase the rate. Their state determines the state of, and hence the amount of rate increase at, the RL.

It is very important to note that the RL goes to HAI only after at least 500 frames have been sent and 50 msec have passed since the last congestion feedback message was received. This doubly ensures that aggressive rate increases occur only after the RL provides the network adequate opportunity (in frames sent for possible sampling) for sending rate decrease signals should there be congestion. This is vital to ensure the stability of the algorithm, and while optimizations can be performed to improve its responsiveness, in the interests of stability and simplicity, the baseline simulation (Alizadeh [B1]) took no further optimization steps.

30.3 Congestion Controlled Flow

A Congestion Controlled Flow (CCF) consists of all of the frames passing through a single Flow queue (31.2.2.1) in an originating end station. Examples of CCFs include, but are not limited to:

- a) The frames carrying data for a single User Datagram Protocol (UDP, IETF RFC 768, STD0006) connection.
- b) All of the frames generated by a particular process running in an end station.
- c) All of the frames being transmitted by an end station that produce the same value when a hash computation is performed on the source and destination IP addresses, UDP vs. TCP selection, and source and destination UDP/TCP port numbers of the IP packets carried by those frames.
- d) All of the frames passing through an RP that have the same destination_address parameter.
- e) All of the frames being transmitted by an end station.
- f) All of the frames passing through a Bridge Port sharing identical source_address, destination_address, and priority parameters.

- g) All of the frames leaving a router that contain IP packets, and have identical values for their source and destination IP addresses, UDP vs. TCP selection, and source and destination UDP/TCP port numbers.
- h) All of the frames with a certain value for the priority parameter.

All of the frames in a given CCF are sent with the same priority value, which is a CNPV (30.4).

Not all frames transmitted by an end station with an RP need be CCFs. Frames not belonging to CCFs bypass the Flow queues and RPs.

30.4 Congestion Notification Priority Value

A Congestion Notification Priority Value (CNPV) consists of one value of the priority parameter such that all of the bridges' and end stations' ports in a Congestion Notification Domain are configured to assign frames at that value to the same CP and/or an RP. Different CNPVs correspond to classes of applications, or even single applications, such as interprocess communications or disk storage data, that have different requirements for such network resources as latency or bandwidth. Since there are eight values for the priority parameter, a single port in a Virtual Bridged Network can support as many as seven CNPV values; at least one value is required for best-effort traffic.

30.5 Congestion Notification Tag

An end station may add a Congestion Notification Tag (CN-TAG) to every frame it transmits from a CCF. The CN-TAG contains a Flow Identifier (Flow ID) field. The destination_address, Flow ID, and a portion of the frame that triggered the transmission of the CNM are the means provided by this standard by which a station can determine to which RP a CNM applies. How the station makes that determination is beyond the scope of this standard. The reasons for adding a CN-TAG include the following:

- a) An end station can identify a particular flow within a CCF that triggered the CNM.
- b) It can be simpler for the end station to use a Flow ID, rather than to parse the returned mac_service_data_unit of the original triggering frame, in order to identify either the RP or the particular flow.
- c) Depending on the application, e.g., if fragmented IP datagrams are transmitted, it can be impossible to determine either the RP or the particular flow based solely on a returned mac_service_data_unit.

An end station that has only a single RP per CNPV, or that is able to identify the RP on the basis of the returned portion of the triggering frame, can omit transmitting the CN-TAG. The CP always returns a CN-TAG in a CNM with the triggering frame's Flow ID, or with a 0 Flow ID, if the triggering frame has no CN-TAG. The fixed format for the CNM minimizes the effort required of the CP to generate the CNM.

30.6 Congestion Notification Domain

In order for congestion notification to successfully control congestion in a Virtual Bridged Network, the managed objects controlling the congestion notification state machines in the bridges and end stations in that network have to be configured with values that are appropriate to the characteristics of the CCFs generated by the applications that expect congestion controlled services. For example

- a) If frames that were not originated from an RP can enter a CP experiencing congestion, then the CNMs generated by that CP upon receipt of those frames cannot correct the problem, and congestion notification cannot operate correctly.
- b) Congestion notification cannot operate correctly if a CP's configuration is inappropriate for the CCFs passing through it, or if priority values are regenerated in a manner that moves frames in and

out of CNPVs.

- c) Frames transmitted from an end station with a CN-TAG cannot be understood by an end station that is not congestion aware.

Congestion aware bridges therefore construct a Congestion Notification Domain (CND), within which a particular CNPV is supported. A CND is a connected subset of the bridges and end stations in a Virtual Bridged LAN that are appropriately configured to serve a particular CNPV. This standard does not assume that every bridge in a Virtual Bridged Network will be capable of congestion control, does not assume that every capable bridge will be so configured, and does not assume that the subset of bridges in a Virtual Bridged Network forming a CND that serves one CNPV will be the same subset that form a CND serving another CNPV.

CNDs can be created by configuring the bridges and end stations in a network, or they can be created automatically, using an additional Type Length Value element, the Congestion Notification TLV (D.2.8), that this standard adds to the IEEE Std 802-2004 Link Layer Discovery Protocol. Either way, every Bridge Port knows, for each CNPV, whether its neighbor(s) are or are not all configured with a matching CNPV.

NOTE—If CND boundaries are configured individually, rather than through LLDP, and there is an error in the configuration, there is a risk that frames not originating from RPs will pass through a CP, or that RP-originated frames will pass through a congested Port that has no CP. Either case can result in congestion that cannot be alleviated, with consequent excessive discarding of RP-originated data frames. In addition, such misconfiguration can result in a congestion aware end station being unable to communicate with a congestion unaware end station, because the latter is receiving CN-TAGs that it does not understand.

Making each CND/CNPV independent, rather than making a single determination of neighbor capabilities based on whether or not all CNPVs are configured the same for all Ports attached to the LAN, allows a CND to be added to or removed from a network already using another CND, without disrupting the operation of the first CND.

30.7 Multicast data

Although control of multicast streams is not an object of this standard, and no simulations or experiments involving multicast data were utilized in the writing of this standard, the transmission of multicast data through a CCF is not prohibited. Some applications primarily send unicast data, but send occasional multicast frames for purposes such as discovery. Transmission of multicast data has been allowed in order to allow well-designed applications using a very small percentage of multicast traffic to operate on a CNPV.

Multicast streams are often sent to multiple destinations along multiple paths through the network, and a unicast stream normally travels along a single path. Therefore, a multicast stream is more likely to encounter any congested ports that exist in a network than is a unicast stream. The consequences of sending multicast streams on a CCF include the following:

- a) The RP could limit the CCF to the rate of the slowest (most congested) path taken by the multicast.
- b) The chances of unintended fate sharing are higher in a CCF that included multicast streams. That is, a multicast stream is more likely to unnecessarily slow down a unicast stream sharing the same CCF than one unicast stream is likely to slow down another.

30.8 Congestion notification and additional tags

The transmission of a CNM from a CP to an RP makes the CP and RP peer entities in the sense of 6.1. An example of this peering relationship for a VLAN Bridged Network is illustrated in Figure 30-5, with the CP-RP relationship shown by a heavy dashed line.

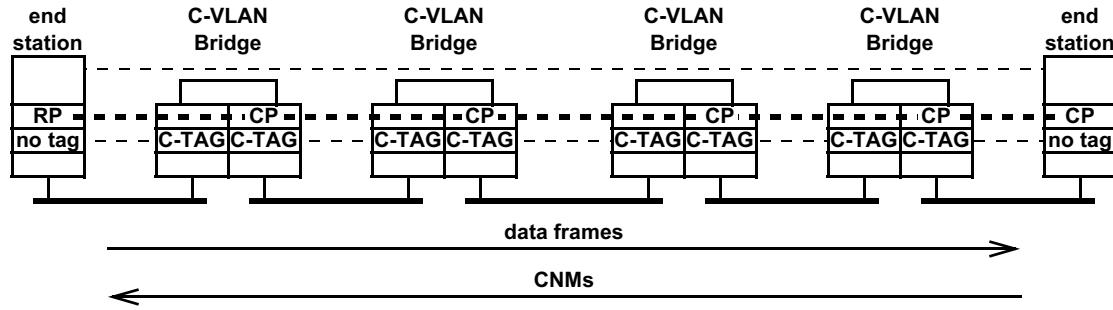


Figure 30-5—CP–RP peering in VLAN Bridged Network

Provider Backbone Bridges (Clause 25) could be used in a data center environment, in order to reduce the number of MAC addresses that have to be learned in the core of the network. The I-component (5.7) of a Provider Backbone Bridge adds and removes I-TAGs (9.7) on data frames. However, as illustrated in Figure 30-6,⁴² adding an I-TAG to a data frame by an I-component impairs the ability of a CP in the core of that network to return a CNM that will be meaningful to the originating end station. The following two difficulties follow from the fact that a CP in a B-VLAN Bridge is not a peer of the RP in an end station:

- The Backbone Core Bridge CP cannot parse the I-TAG and C-TAG, which is necessary in order to find the RP's MAC address and the data frame's CN-TAG, in order to build an I-TAG-encapsulated, CNM addressed to the RP.
- There is no MAC address available to the Backbone Core Bridge CP that would be a valid source _address in the CNM when received by the RP.

In order to provide the benefits of PBBNs or other hierarchical bridging technologies to the data center environment, an interworking function (32.16) is defined to allow core CPs to peer with end stations.

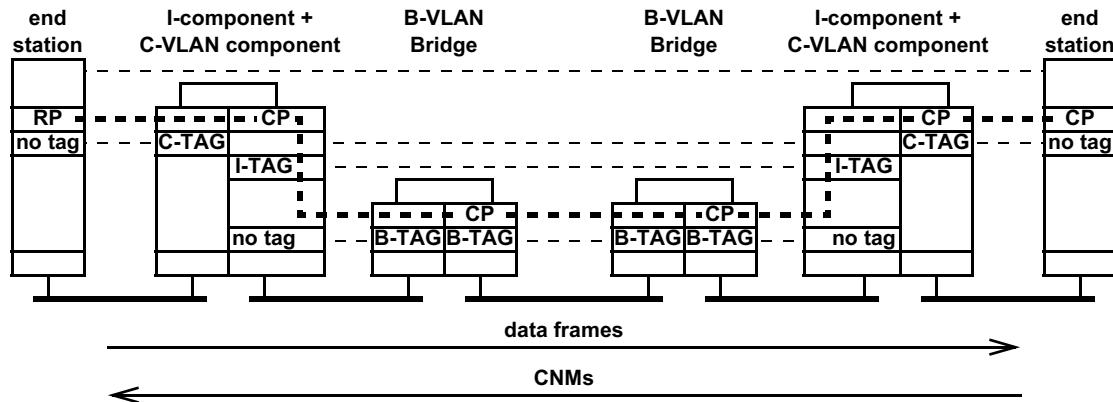


Figure 30-6—CP–RP peering in Provider Backbone Bridged Network

⁴²In the configuration chosen for illustration for the Data Center environment in Figure 30-6, each S-VLAN component of an I-components is, in effect, a Provider Bridge (5.10). That is, each Customer Network Port in an I-component is virtualized and multiplied, and connected to a corresponding virtual Provider Edge Port of a C-VLAN component offering a single Customer Edge Port facing the end station. This offers a C-TAG interface to the end station, without incurring the burden of an additional layer of S-TAGs.

31. Congestion notification entity operation

This clause specifies the following:

- a) The architecture of the CP in the Forwarding Process in a congestion aware bridge component or end station (31.1).
- b) The architecture of a CP and an RP in a congestion aware end station (31.2).

NOTE—Clause 30 introduces the principles of congestion notification operation and the network architectural concepts that support it. Clause 32 specifies the protocols operated by entities defined in this clause. Clause 33 specifies the encoding of the PDUs used by congestion notification.

The models of operation in this clause provide a basis for specifying the externally observable behavior of congestion notification, 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.

31.1 Congestion aware Bridge Forwarding Process

Figure 8-10 illustrates the details of the Bridge Forwarding Process. Figure 22-1 shows a different view of those details, rearranged so that those entities of the Forwarding Process that are concerned only with a single Bridge Port can be put in the proper relationship to entities such as Connectivity Fault Management shims. Figure 31-1 shows only the queuing functions of Figure 22-1, as modified for a congestion aware bridge. Two new entities are added for the congestion aware bridge: the Congestion Point (31.1.1) and the Congestion Point ingress multiplexer (31.1.2).

31.1.1 Congestion Point

As described in 32.8 and 32.9, based on the state of its internal variables, and a frame given the CP by the Queuing frames entity (8.6.6) in an EM_UNITDATA.request (parameters) (32.9.3), a CP either inserts the frame into its attached queue or discards the frame, and can generate a Congestion Notification Message (CNM), based on the frame. Discarded frames are counted in the variable cpDiscardedFrames (32.8.12). Frames successfully queued are counted in the variable cpTransmittedFrames (32.8.13). Any CNM generated is passed to the Congestion Point ingress multiplexer (31.1.2, Figure 31-1) or Flow multiplexer (31.2.4, Figure 31-2) for transmission back to the RP that sourced the frame.

Congestion notification does not operate correctly if frames not belonging to a Congestion Controlled Flow (CCF) are allowed to pass through a CP. The CND operations (32.1) help to ensure that the only frames passing through a CP belong to Congestion Controlled Flows (CCFs).

The CP shall remove the CN-TAG from every frame passing through it with a priority value (a CNPV) for which the port on which the CP resides is acting in cptEdge or cptInterior CND defense mode.

The functions of Figure 22-1 could be implemented in different ways than that illustrated, without altering the externally observed behavior of the conformant equipment. For example, the CP could pass a copy of the output frame triggering a CNM to a function in the higher layer interfaces of a bridge, which in turn, could generate the CNM. However, in order to provide a management interface that is useful over a wide range of implementations, the following provisions are made:

- a) CNMs generated on a Bridge Port are not counted in the Frames Received (12.6.1.1.3:a) or Octets Received (12.6.1.1.3:b) counters.
- b) CNMs generated on one Bridge Port are counted, as are ordinary data frames, on the Bridge Ports (if any) on which they are output.

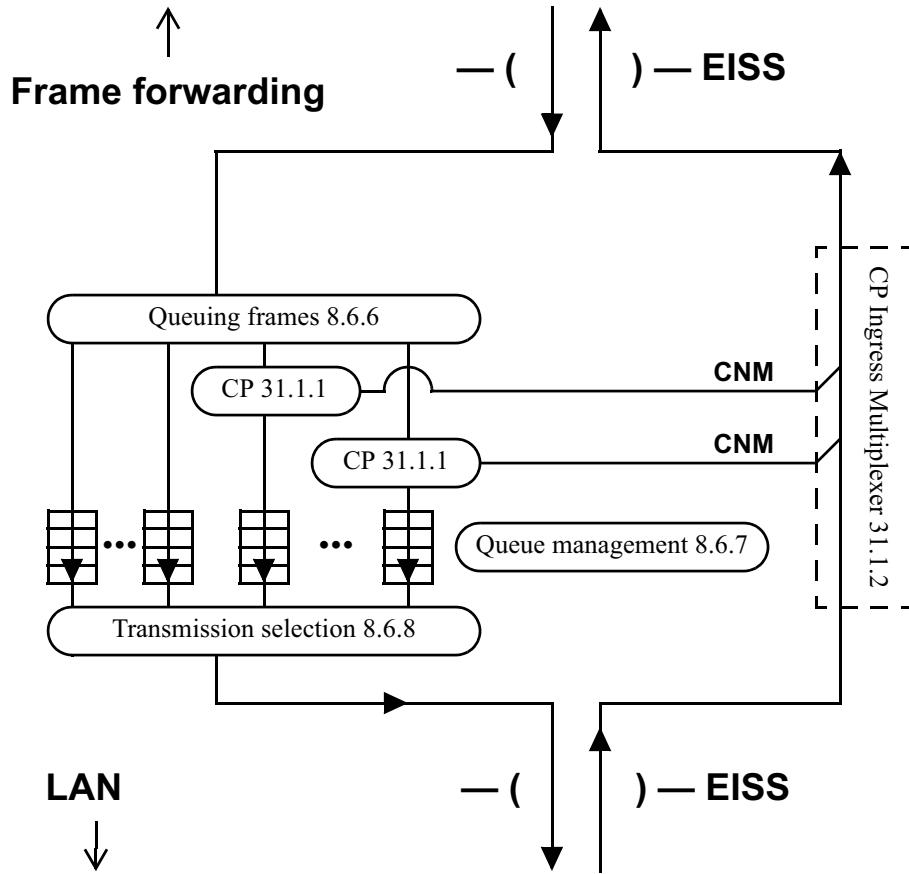


Figure 31-1—Congestion Points and congestion aware queues in a bridge

If a bridge supports the dot1agCfmVlanTable or the ieee8021CfmVlanTable, its CPs transmit each CNM to the Primary VID associated with the vlan_identifier of the frame triggering the CNM's transmission.

31.1.2 Congestion Point ingress multiplexer

Inserts the CNMs generated by the Congestion Points (31.1.1) among the frames received from the LAN.

31.2 Congestion aware end station functions

Figure 31-2 illustrates the architecture of the queue functions of a congestion aware end station. These functions offer a service to higher layers through a single instance of the ISS or EISS, and utilize an instance of the ISS or EISS to connect to the lower layers. The Output flow segregation function is presumed to exercise control over the higher layers' ability to present frames for transmission through the ISS or the EISS, and the Reception selection function to control the flow of frames to the higher layers, by means of additional parameters local to this instance of the ISS or the EISS and/or locally significant LMI parameters (6.1.3), neither of which are specified by this standard.

In Figure 31-2, the outbound queue block, in the lower left of Figure 31-2, is exactly the same as the output queues in a Bridge Port that is not congestion aware, as illustrated in Figure 22-1.

The remaining entities illustrated in Figure 31-2 are those that are peculiar to a congestion aware end station, and are further described in the following subclauses.

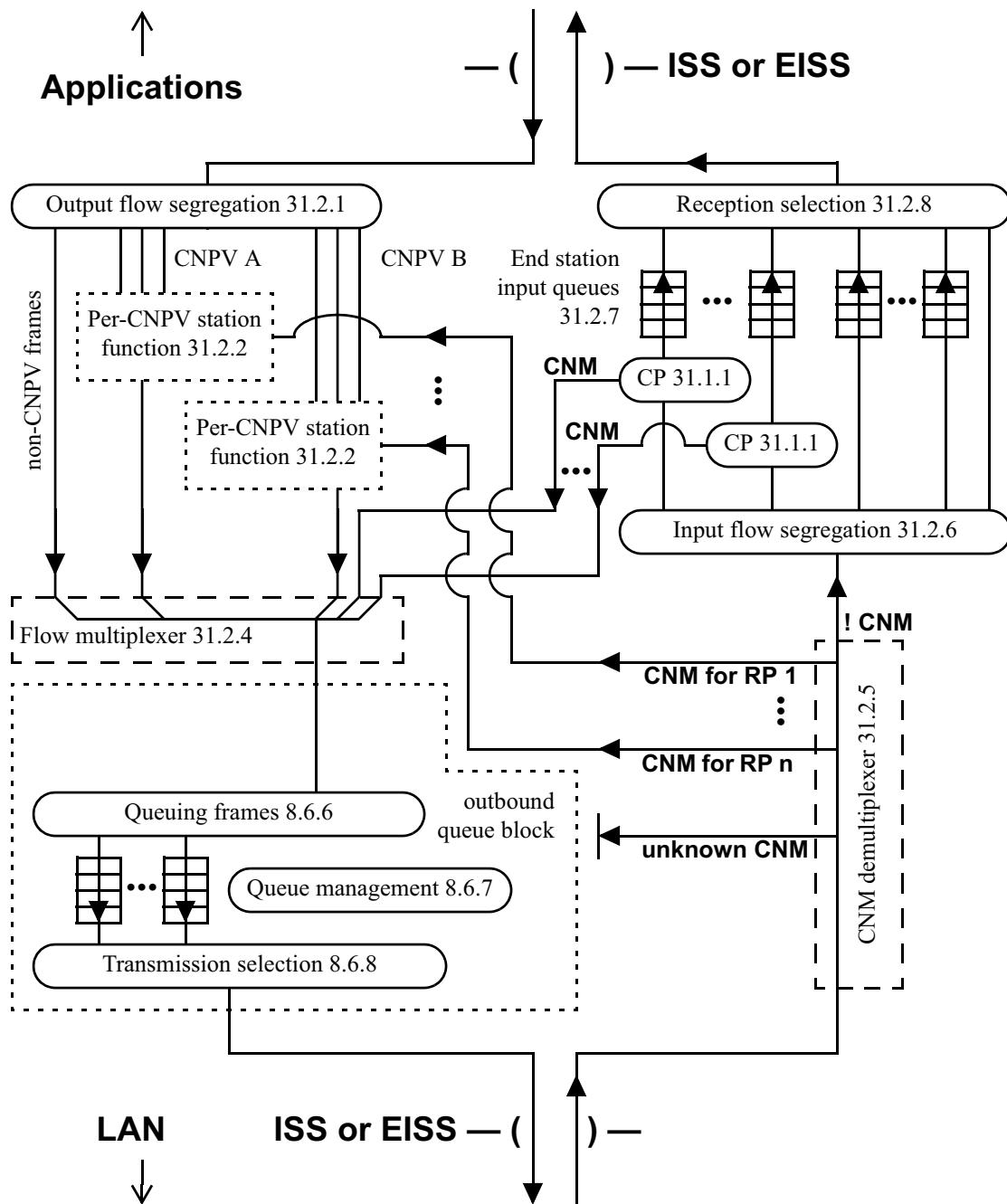


Figure 31-2—Congestion aware queue functions in an end station

31.2.1 Output flow segregation

For each frame received from the upper layers for transmission, the Output flow segregation entity

- Assigns the frame a priority parameter based on the priority presented from the upper layer and/or upon other criteria, unspecified by this standard.
- Determines, using priority value and the Flow Select Database (31.2.3), to which Flow queue, if any, the frame is to be assigned.

- c) If the frame is not assigned to any Flow queue (the frame's priority is not a CNPV), passes the frame directly to the Flow multiplexer entity (31.2.4).
- d) If the frame is assigned to a particular Flow queue (the frame's priority is a CNPV), enqueues the frame in the selected Flow queue.
- e) For frames that are assigned to a Flow queue, depending on the state of the Flow queue to which the frame is assigned, and through means unspecified by this standard, can influence the higher layers' presentation of further frames to the Output flow segregation entity.

The Output flow segregation entity never discards a frame. Every frame presented to it is either passed to the Flow multiplexer entity (31.2.4) or stored in a Flow queue, as described previously. This standard assumes that the Output flow segregation entity can apply means, unspecified by this standard, to prevent the upper layers from offering a frame that should be stored in a Flow queue, but for which the Flow queue has insufficient resources.

The default configuration for an end station shall have a single Flow queue per CNPV. An end station may support additional Flow queues, or support more than one CNPV with a single Flow queue.

NOTE—Requirements for a default configurations do not imply that explicit action via SNMP is required to change from the default. For example, managed objects in an end station can be adjusted by an application program.

If the Output flow segregation function in an end station can change its method for assigning frames to Flow queues, it shall not do so in a manner that impairs the ability of any compliant CP implementation to throttle the data from the sourcing end station, and shall not materially increase the likelihood of misordered frame transmissions in a manner that adversely impacts the higher layers' functions.

31.2.2 Per-CNPV station function

There is one Per-CNPV station function on each Port of a congestion aware end station for each priority value that is a CNPV. All of the frames passing through a Per-CNPV station function have the same CNPV priority parameter value.

As illustrated in Figure 31-3, each Per-CNPV station function is composed of the following entities:

- a) One or more Flow queues (31.2.2.1).
- b) One or more Reaction Points (31.2.2.2), each associated with exactly one Flow queue, each comprising
 - 1) A Reaction Point State Machine (31.2.2.3) for each Reaction Point.
 - 2) A Rate Limiter (31.2.2.4) for each Reaction Point.
- c) One Flow Selection function (31.2.2.5).

31.2.2.1 Flow queue

A Flow queue is a first-in-first-out queue of frames, all for the same CNPV. An Output flow segregation entity selects the Flow queue into which a given frame is inserted, and an RP determines when a frame is removed from a Flow queue.

NOTE—The Flow queue is a notional convention used to unambiguously describe the behavior of the RP. An end station can implement the Flow queue without explicitly storing frames in a queue, based on its knowledge of the state of the higher-layer applications supplying the frames.

31.2.2.2 Reaction Point

A single Reaction Point (RP) is associated with each Flow queue. An RP is composed of a Reaction Point State Machine (31.2.2.3) and a Rate Limiter (31.2.2.4).

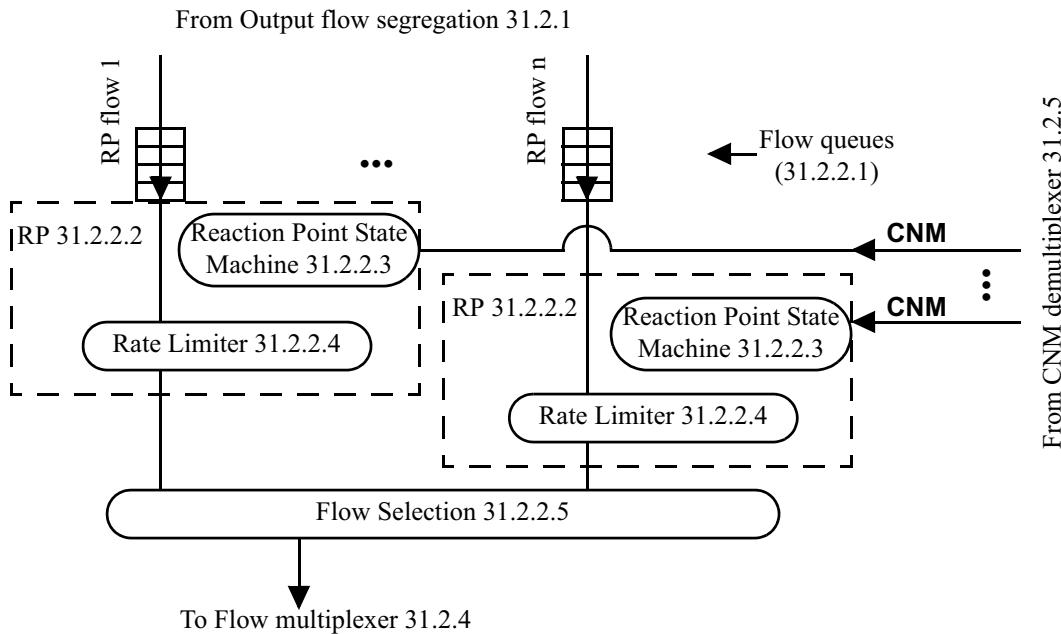


Figure 31-3—Per-CNPV station function

31.2.2.3 Reaction Point State Machine

A single Reaction Point State Machine is associated with each Flow queue. As described in detail in Clause 32, the Reaction Point State Machine consists of a timer (32.12), variables (32.10, 32.11, and 32.13), procedures (32.14), and a state machine (32.15) that determine the value of the rpLimiterRate variable (32.13.8), which, via the Rate Limiter (31.2.2.4), controls the transmission of frames from the Flow queue. The state machine and variables, in turn, are controlled by the frames input, managed objects (12.21), the Output flow segregation entity (31.2.1), and the CNMs received from the CNM demultiplexer (31.2.5).

31.2.2.4 Rate Limiter

A single Rate Limiter is associated with each Flow queue. The Rate Limiter enforces the output rate, rpLimiterRate (32.13.8), determined by the Reaction Point State Machine (31.2.2.3). The octets in the frames passed from the Rate Limiter to the Flow Selection function (31.2.2.5) are counted in rpByteCount (32.13.2). The details of the operation of the Rate Limiter are not specified by this standard, but the measurable output rate is specified.

For a given measurement period T, we define R_T as the integral of rpLimiterRate over T, and L_T as the number of octets actually output to the LAN, including preambles, inter-frame gaps, etc., stemming from frames that pass through that RP's Rate Limiter. An end station shall ensure that:

$$L_T \leq (1.05 \times R_T) + (16 \text{ maximum-sized frames}) \quad (1)$$

for the two values of T = one second and T = 1 million bits divided by the physical line rate (in bits per second), where any number of simultaneous measurements can start at any time.

NOTE 1—The rate computations and variables [Equation (1), 32.13.6, 32.11.6, etc.] include all of the octets associated with a frame on the LAN, including preambles, inter-frame gaps, etc., and thus reflect actual physical data rates. However, the resource actually protected by congestion notification is octets in buffers, not octet times on wires. Per-frame overhead on the LAN is added to serve as a reasonable approximation, made at the RP, to the actual size of the

frame in the Bridges' or end stations' buffers. The extra 5% margin in Equation (1) is provided because this approximation is inexact.

NOTE 2—Although the simulations have been studied with a burst of one maximum-sized frame, this standard allows the end stations to send data in bursts of 16 maximum-sized frames, primarily to ease implementations. Larger bursts of data have a negative impact on the overall performance of the Quantized Congestion Notification protocol (QCN) algorithm, and can lead to lower link utilization.

31.2.2.5 Flow Selection

The Flow Selection function decides from which Flow queue in its Per-CNPV station function, if any, a frame will be offered to the Flow multiplexer. Whenever the output queue that receives frames from the Flow Selection function (via the Flow multiplexer) is full, the Flow Selection function shall not present any frame to the Flow multiplexer; this ensures that no frames in CNPVs will be discarded in the Port due to a lack of room in the output queue.

By means unspecified in this standard, the Flow Selection function selects frames to pass to the Flow multiplexer from the uncontrolled Flow queues (31.2.2.1) and Rate Limiters (31.2.2.4). Whenever a frame is passed from a Rate Limiter through the Flow Selection function to the Flow multiplexer, the Flow Selection function calls *TransmitDataFrame* (32.14.3) to update the Reaction Point variables (32.13). If the port's neighbor is ready to receive frames with CN-TAGs for the frame's CND (defense mode is *cptInteriorReady*), the Flow Selection function may insert a CN-TAG (30.5) at the beginning of the frame's *mac_service_data_unit*, and if not (any CND defense mode other than *cptInteriorReady*), it shall not insert a CN-TAG. (See 33.2 for a further explanation of CN-TAG insertion.)

31.2.3 Flow Select Database

The Flow Select Database is used by the Output flow segregation (31.2.1) entity to determine to which Flow queue (31.2.2.1) a given frame is assigned.

The means by which the Flow Select Database selects a Flow queue is unspecified by this standard.

31.2.4 Flow multiplexer

The Flow multiplexer merges the frames from the Per-CNPV station functions' Flow Selection functions (31.2.2.5), the frames bypassing the Per-CNPV station functions, and the CNM frames from the Congestion Points (31.1.1), and delivers the frames to the Queuing frames (8.6.6) entity in the end station's output path.

31.2.5 CNM demultiplexer

The CNM demultiplexer identifies CNMs received from the LAN, uses the Flow Identifier (33.2.1) and/or Encapsulated priority (33.4.7) to determine to which RP (31.2.2.2) each CNM is intended, and directs them to the correct RPs. This standard does not specify whether or how the end station can use the Flow Identifier to further identify one or more individual flows within a Flow queue and its associated RP. If a CNM is received that has no CN-TAG, or a CN-TAG whose Flow ID is unknown to the RP, the CNM is discarded. All non-CNM frames are passed to the Input flow segregation entity (31.2.6).

31.2.6 Input flow segregation

For each frame received from the lower layers for reception, the Input flow segregation entity:

- a) If the frame's priority is not a CNPV, decides, in an unspecified manner, whether to pass the frame to an unspecified queue, pass it to the higher layers, or discard it.
- b) If the frame's priority is a CNPV, passes the frame to a CP (31.1.1), selected by unspecified means, for processing.

NOTE 1—For example, an end station might support a single CP, or support one CP per CNPV, per VID, per application, or per data flow. Neither these nor any choices for CP selection are required by this standard.

The Input flow segregation entity never discards a CNPV frame. Every CNPV frame presented to it is passed to a CP, as described previously.

NOTE 2—This does not mean that a station cannot drop CNPV frames. See 31.2.7.

31.2.7 End station input queue

The number and operation of end station input queues and the method used to drain these queues (i.e., to select and transfer frames to the receiving higher layer applications) are not specified by this standard. However, the procedures performed by a CP (32.9) require that its associated end station input queue exist, and possess certain operational parameters (32.8).

In the unfortunate event that a misconfiguration or an unfortunate sequence of frame transmissions results in more CNM frames arriving at a station than it can process, an end station input queue can fill up, and discard a frame presented to it by its CP or by the Input flow segregation (31.2.6) entity.

31.2.8 Reception selection

The Reception selection entity selects frames for delivery to the upper layers. It takes the place of the Transmission selection entity (8.6.8) of a congestion aware bridge. The selection algorithm depends on interactions between the applications receiving the data, the operating system supporting those applications, and the received frames that are not specified by this standard. The Reception selection entity is shown in Figure 31-2 for completeness, to indicate that frames enqueued by the Queuing frames entity (8.6.6) are, in fact, removed from the queues.

If the first octets of the `mac_service_data_unit` of a frame selected by the Reception selection entity are a CN-TAG, Reception selection removes the CN-TAG from the `mac_service_data_unit` before passing the frame to the higher layers. (See 33.2 for a further explanation of CN-TAG removal.)

32. Congestion notification protocol

Congestion-aware systems can participate in the congestion notification protocols specified in this clause, namely:

- a) Congestion Notification Domain (CND) operations (32.1).
- b) The variables controlling congestion notification at the end station or bridge component (32.2) and Port and Congestion Notification Priority Value (CNPV) levels (32.3),
- c) The variables (32.4), procedures (32.5), and state machine (32.6) that control the defense of a CND.
- d) The congestion notification protocol (32.7).
- e) The variables (32.8) and procedures (32.9) that define a CP.
- f) The variables associated with all (32.10), or a subset of (32.11), the RPs on a given Port and CNPV.
- g) The timer (32.12), variables (32.13), procedures (32.14), and state machine (32.15) that define the operation of a single a RP.
- h) The processing of CNMs by a congestion notification and encapsulation interworking function (32.16).

NOTE—Clause 30 introduces the principles of congestion notification operation and the network architectural concepts that support it. Clause 31 breaks down the congestion notification entities into their components. Clause 33 specifies the encoding of the PDUs used by congestion notification.

32.1 Congestion Notification Domain operations

As described in 30.6, a CND is a connected subset of a Virtual Bridged Network configured to support a particular Congestion Notification Priority Value (CNPV). This standard provides a means whereby a VLAN-aware Bridge or end station can recognize the like-configured ports of a CND (32.1.1), and defend its CND against incoming frames from outside the CND.

32.1.1 Congestion Notification Domain defense

Unless every bridge along a path between two congestion aware end stations using a particular CNPV is configured for congestion notification (belongs to a CND), and unless the bridges ensure that frames not in CNPVs use different queues than the queues used by those two end stations, those end stations will not accrue the advantages that congestion notification offers. Therefore, steps must be taken at the boundaries of a CND to protect both the CND and the network outside the CND.

Every frame received on a Port passes through the ISS Support by specific MAC procedures (6.7), which can change the priority parameter of the received frame, using the Port's Priority regeneration table, Table 6-5. This table can be controlled by a managed object (12.6). In addition, in order to prevent frames not in CNPVs from entering CP-controlled queues, each Port's Priority regeneration table can be modified.

If a Port's neighbor is known to also be configured for a particular CNPV, the entry in the Port's Priority regeneration table for that CNPV is ignored, and the priority is never changed on input. The bridge component or end station connected to that Port can be trusted to use that CNPV. If the Port's neighbor is known to not be configured for a particular CNPV, the entry in the Port's Priority regeneration table for that CNPV shall be overridden to translate the CNPV to an alternate non-CNPV value. The neighboring system is not trusted, so the priority values of any frames carrying the CNPV's value are changed. Altering the priority values of received frames defends the CND from frames not originating from an RP. Similarly, if a CNPV is configured on any Port, then on that Port, the Port's Priority regeneration table shall be overridden to prevent any other priority value from being remapped into that CNPV.

If a port has multiple neighbors, or if the neighbor on a Port is not congestion aware, then the CN-TAGs shall be removed from any frames transmitted on a CNPV. This ensures that the benefits of congestion notification can be provided as far as possible along a path from a congestion aware source to a

noncongestion aware destination. Furthermore, Congestion Notification Tags (CN-TAGs) shall be removed from any frames transmitted from a CNPV toward a system that is congestion aware, but is still remapping the CNPV to a non-CNPV priority.

Thus, for each priority value, the Congestion Notification Domain defense mode can take one of the following four values, whether derived from LLDP or set by managed variables:

cptDisabled: The congestion notification capability is administratively disabled for this priority value and port. This priority is not a CNPV. The priority regeneration table (6.9.4) controls the remapping of input frames on this port to or from this priority. CN-TAGs are neither added by an end station nor stripped by a bridge.

cptEdge: On this port and for this CNPV, the priority parameters of input frames are remapped to an alternate (non-CNPV) value, and no priority value is remapped to this CNPV, regardless of the priority regeneration table. CN-TAGs are not added by an end station and are removed from frames before being output by a bridge.

NOTE—The cptEdge CND defense mode is optional for an end station. See 32.4.9.

cptInterior: On this port and for this CNPV, the priority parameters of input frames are not remapped to another value, and no priority value is remapped to this CNPV, regardless of the priority regeneration table. CN-TAGs are not added by an end station, and are removed from frames before being output by a bridge.

cptInteriorReady: On this port and for this CNPV, the priority parameters of input frames are not remapped to another value, and no priority value is remapped to this CNPV, regardless of the priority regeneration table. CN-TAGs can be added by an end station and are not removed from frames by a bridge.

The determination of the operational state of a CNPV on a given port is determined by variables and by LLDP (32.1.1, 32.2.1, 32.3.7, 32.3.4, 32.4.1, 32.4.3).

Taken together, these capabilities ensure that as long as the default values of the managed variables controlling CND defense are not overridden with incorrect values:

- a) A data frame received by an end station on a CNPV has been governed by CN along its entire path from the transmitting end station.
- b) Uncontrolled data streams sourced by an end station that is congestion unaware cannot pass through a CP, with a resultant excessive loss of congestion aware frames.
- c) A data frame carrying a CN-TAG cannot be delivered to a system that is congestion unaware, and thus be unreadable by that system.

It is therefore recommended that the network administrator allow CND recognition to operate automatically (32.1.2).

CND defense operates on each LAN separately; it does not ensure the existence of an end-to-end congestion aware path from end station to end station through a VLAN Bridged Network. Misconfiguration, the presence along the path of a congestion unaware bridge, or the activation of a new bridge or LAN (temporarily) can cause frames to exit a CND, and hence the following to occur:

- d) Data frames can be transmitted by a congestion aware end station on a CNPV, then be uncontrolled for congestion along some part of their path, and be delivered to an end station on a non-CNPV priority value with no CN-TAG.
- e) If a congestion aware end station transmits data frames with CN-TAGs, and is connected to a congestion unaware end station along a path consisting entirely of congestion unaware or

misconfigured bridges, the receiving end station will be unable to understand the CN-TAG, and thus unable to process the data frames.

- f) If a system generates CNMs on a CNPV (not recommended, see 32.2.2), they can be delivered on a non-CNPV priority value with no CN-TAG.

32.1.2 Automatic Congestion Notification Domain recognition

Congestion notification can use the Link Layer Discovery Protocol (LLDP), defined in IEEE Std 802.1AB, to advertise the CNPVs supported on a bridge's or end station's port, to determine whether the other bridges' or end stations' ports connected to the same LAN are configured for those same CNPVs, and to control the operation of the CND defenses (32.1.1).

Multiple instances of LLDP can be run on a single port. Each instance's LLDPDUs use a different destination_address, and therefore can reach different distances through a Virtual Bridged Network, depending on the types of the devices in that network and their connectivity. Since any bridge that is not congestion aware can spoil the ability of the Virtual Bridged Network to minimize frame loss, the selection of which LLDP instance is used for Congestion Notification Domain defense defaults to the Nearest Bridge Address (01-80-C2-00-00-0E). This instance minimizes the reach of LLDP, and hence the risk that a noncongestion aware bridge is present.

As described in D.2.8, a single CN TLV can be inserted into the LLDPDUs transmitted from any bridge or end station port that is configured to support a CNPV. The CN TLV carries two fields that together provide for automatic control the of the CND defenses:

- a) Eight per-priority CNPV indicators (D.2.8.3), one per priority level, indicating whether each priority is or is not a CNPV on this port; and
- b) Eight per-priority Ready indicators (D.2.8.4), one per priority level, indicating whether the priority remap defenses are disabled on this port.

For every CNPV configured on a port, if LLDP finds exactly one remote neighbor, and if that neighbor's database in the port's LLDP table carries a CN TLV (D.2.8) with that CNPV's bit set in the per-priority CNPV indicators, then that CNPV is known to be configured on the remote system's port. This condition is summarized by the cnpdRcvdCnpv variable (32.4.11). Similarly, the cnpdRcvdReady variable (32.4.11) indicates whether or not the remote neighbor has or has not turned off its priority remapping defense. The read-only managed object cnpdAutoDefenseMode (32.4.3) indicates the results of the LLDP.

32.1.3 Variables controlling Congestion Notification Domain defense

In order to control CND defense, CN component managed objects (12.21.1), CN component priority managed objects (12.21.2), CN Port priority managed objects (12.21.3), and Congestion Point managed objects (12.21.4) override and are overridden by each other as follows:

- a) Altering the CN component managed object (12.21.1) or CN component priority managed object (12.21.2) does not alter the values of any of the other managed objects (12.21.3, 12.21.4, 12.21.6).
- b) If cngMasterEnable (32.2.1) is FALSE, all congestion notification activity is suppressed; the Priority Regeneration Table (6.9.4) operates normally, and CNMs, CN-TAGs, and LLDP CN TLVs are never generated and are always ignored on receipt. If TRUE, the other managed objects control the operation of congestion notification.
- c) The Port/priority and component objects override each other as shown in Table 32-1 and Table 32-2. In each case, if the Port/priority object contains the specified value, the component or component/priority object controls the specified function.

Table 32-1—LLDP instance selection managed object overrides

(Per-port per-priority) cnpdLldpInstanceChoice 32.4.4	(Component per-priority) cnccLldpInstanceChoice 32.3.6	Congestion Notification TLV is sent/received	LLDP instance is selected by
cnlNone	any	No	None selected
cnlAdmin	any	Yes	(Per-port per-priority) cnpdLldpInstanceSelector 32.4.5
cnlComponent	cnlNone	No	None selected
cnlComponent	cnlAdmin	Yes	(Component per-priority) cnccLldpInstanceSelector 32.3.7

Table 32-2—CND defense mode selection managed object overrides

(Per-port per-priority) cnpdDefModeChoice 32.4.1	(Component per- priority) cnccDefModeChoice 32.3.1	CND defense mode is selected by	Alternate priority is selected by
cpcAdmin	any	(Per-port per-priority) cnpdAdminDefenseMode 32.4.2	(Per-port per-priority) cnpdAlternatePriority 32.4.6
cpcAuto	any	(Per-port per-priority) cnpdAutoDefenseMode 32.4.3	(Component per-priority) cnccAutoAltPri 32.3.3
cpcComp	cpcAdmin	(Component per-priority) cnccAdminDefenseMode 32.3.4	(Component per-priority) cnccAlternatePriority 32.3.2
cpcComp	cpcAuto	(Per-port per-priority) cnpdAutoDefenseMode 32.4.3	(Component per-priority) cnccAutoAltPri 32.3.3

32.2 CN component variables

Every congestion aware end station or bridge component has a set of CN component variables to control the overall operation of congestion notification. All of these variables are included in the CN component managed object (12.21.1). Variables in this section marked, “CP only” are not required for an end station that does not support CPs. The CN component variables include the following:

- a) cngMasterEnable (32.2.1)
- b) cngCnmTransmitPriority (32.2.2)
- c) cngDiscardedFrames (32.2.3)
- d) cngErroredPortList (32.2.4)

32.2.1 cngMasterEnable

A Boolean value specifying whether congestion notification is enabled in this bridge component or end station.

32.2.2 cngCnmTransmitPriority

(CP only) The priority value to be used when transmitting CNMs from this bridge component or end station (default 6) (32.9.4:h).

NOTE—If cngCnmTransmitPriority is itself a CNPV, then it is possible that another CP could generate a CNM in response to a CNM sent by this Port. Although this is not desirable, the fact that CNMs are sent in response to only a sampling of frames means that the network will not fail due to an excessive number of CNMs.

32.2.3 cngDiscardedFrames

(CP only) The total number of frames discarded from full CP queues. This is the total of the values of all cpDiscardedFrames (32.8.12) variables for this end station or bridge component.

32.2.4 cngErroredPortList

(CP only) A list of Ports whose alternate priority values (cnpdAlternatePriority, 32.4.6) specify a CNPV, where 0 indicates that the bridge's alternate priority value (cnnpAlternatePriority, 32.3.2) specifies a CNPV.

NOTE—The alternate priority (cnnpAlternatePriority, 32.3.2, or cnpdAlternatePriority, 32.4.6) for a given CNPV cannot itself be a CNPV. But, because alternate priority values can be set using different managed objects for different CNPVs, invalid configurations can be specified. cngErroredPortList provides the network administrator with a list of Ports that could have invalid configurations.

32.3 Congestion notification per-CNPV variables

An instance of the congestion notification per-CNPV variables exists for each CNPV in a congestion aware end station or bridge component to control CN for all Ports in the end station or bridge component. Variables in this section marked, “CP only” are not required for an end station that does not support CPs. The congestion notification per-CNPV variables include the following:

- a) cnnpDefModeChoice (32.3.1)
- b) cnnpAlternatePriority (32.3.2)
- c) cnnpAutoAltPri (32.3.3)
- d) cnnpAdminDefenseMode (32.3.4)
- e) cnnpCreation (32.3.5)
- f) cnnpLldpInstanceChoice (32.3.6)
- g) cnnpLldpInstanceSelector (32.3.7)

32.3.1 cnnpDefModeChoice

An enumerated value specifying how the CND defense mode and alternate priority for all Ports for this CNPV are to be selected, unless overridden (32.1.3) by the variables in the CND defense per-Port per-CNPV variables (32.4), either one of the following:

- 1) **cpcAdmin:** The Port's default CND defense mode for this priority is controlled by cnnpAdminDefenseMode (32.3.4) and the alternate priority by cnnpAlternatePriority (32.3.2); or

- 2) **cpcAuto:** The Port's default CND defense mode for this priority is controlled by LLDP and the Congestion Notification TLV (cnpdAutoDefenseMode, 32.4.3) and the alternate priority by cncpAutoAltPri (32.3.3).

32.3.2 cncpAlternatePriority

(CP only) An alternate priority value to which this priority value is to be remapped when the Port's CND defense mode is cptEdge for this priority (default value 0). This can be overridden by cnpdAlternatePriority (32.1.3, 32.4.6).

32.3.3 cncpAutoAltPri

An integer indicating the next lower priority value than this CNPV that is not a CNPV in an end station or bridge component, or the next higher non-CNPV, if all lower values are CNPVs (32.1.1).

32.3.4 cncpAdminDefenseMode

An enumerated value controlling the CND defense mode (30.6) for all Ports for this CNPV in this bridge component or end station: cptDisabled (1), cptInterior (2, the default value), cptInteriorReady (3), or cptEdge (4) (32.1.1). This can be overridden by cnpdAdminDefenseMode (32.1.3, 32.4.2).

32.3.5 cncpCreation

An enumerated value controlling the applicability of this CN component priority managed object to the Ports in that bridge component or end station, either one of the following:

- 1) **cncpAutoEnable:** The cnpdDefModeChoice variable (32.4.1) in any CN Port priority managed object created as a result of the creation of this CN component priority managed object or the creation of a Port is given the value cpcComp (3); or
- 2) **cncpAutoDisable:** The cnpdDefModeChoice variable (32.4.1) in any CN Port priority managed object created as a result of the creation of this CN component priority managed object or the creation of a Port is given the value cpcAdmin (1).

32.3.6 cncpLldpInstanceChoice

Determines the method by which the component LLDP instance selector, specifying on which LLDP instance the Congestion Notification TLV is to carry information for this priority, is determined (32.1.1), either one of the following:

- 1) **cnlNone:** if LLDP is not to carry Congestion Notification TLVs on any Port or priority; or
- 2) **cnlAdmin:** If cncpLldpInstanceSelector (32.3.7) governs LLDP instance selection for all Ports and priorities (this is the default value).

This choice can be overridden (32.1.3) on a per-Port per-priority basis by cnpdLldpInstanceChoice (32.4.4).

32.3.7 cncpLldpInstanceSelector

A reference to the LLDP destination address table entry whose LLDPDUs are to carry Congestion Notification TLVs for this bridge component or end station (32.1.1). The use of this variable is controlled by cncpLldpInstanceChoice (32.3.6).

32.4 CND defense per-Port per-CNPV variables

For each Port in a congestion aware end station or bridge component, and for each priority value for which a set of Congestion notification per-CNPV variables exists, there is an instance of the CND defense per-Port per-CNPV variables. These variables control congestion notification on the port for a CNPV, including Congestion Notification Domain defense. Variables in this section marked, “CP only” are not required for an end station that does not support CPs. The CND defense per-Port per-CNPV variables include the following:

- a) cnpdDefModeChoice (32.4.1)
- b) cnpdAdminDefenseMode (32.4.2)
- c) cnpdAdminDefenseMode (32.4.2)
- d) cnpdAutoDefenseMode (32.4.3)
- e) cnpdLldpInstanceChoice (32.4.4)
- f) cnpdLldpInstanceSelector (32.4.5)
- g) cnpdAlternatePriority (32.4.6)
- h) cnpdXmitCnpvCapable (32.4.7)
- i) cnpdXmitReady (32.4.8)
- j) cncpDoesEdge (32.4.9)
- k) cnpdAcceptsCnTag (32.4.10)
- l) cnpdRcvdCnpv (32.4.11)
- m) cnpdRcvdReady (32.4.12)
- n) cnpdIsAdminDefMode (32.4.13)
- o) cnpdDefenseMode (32.4.14)

32.4.1 cnpdDefModeChoice

An enumerated value specifying how the CND defense mode and the alternate priority of the Port for this priority is to be determined, (30.6, 32.1.1, 32.1.1):

- 1) **cpcAdmin:** The Port’s CND defense mode for this priority is controlled by cnpdAdminDefenseMode (32.4.2), and the alternate priority by cnpdAlternatePriority (32.4.6);
- 2) **cpcAuto:** The Port’s CND defense mode for this priority is controlled by LLDP and the Congestion Notification TLV (cnpdAutoDefenseMode, 32.4.3) and the alternate priority by cncpAutoAltPri (32.3.3); or
- 3) **cpcComp:** The Port’s CND defense mode for this priority is controlled by cncpDefModeChoice (32.3.1) and the alternate priority by cncpAutoAltPri (32.3.3) (this is the default value).

32.4.2 cnpdAdminDefenseMode

An enumerated value specifying the CND defense mode of the Port for this priority, if and only if cnpdDefModeChoice (32.4.1) has the value cpcAdmin (1) (30.6, 32.1.1, 32.1.1):

- 1) **cptDisabled:** The CP is disabled on this port for this CNPV. Priority regeneration on input is controlled by the priority regeneration table. This is the same behavior as when cngMasterEnable (32.2.1) has the value FALSE. (This is the default value.);
- 2) **cptInterior:** The priority parameter of frames input are not remapped to or from this priority, and CN-TAGs are not output;
- 3) **cptInteriorReady:** The priority parameter of frames input are not remapped to or from this priority, and CN-TAGs can be output; or
- 4) **cptEdge:** The priority parameter of frames input at this priority are remapped to an alternate value, frames at other priorities are not remapped to this priority, and CN-TAGs are not output.

32.4.3 cnpdAutoDefenseMode

An enumerated value indicating the operating mode: cptInterior (2), cptInteriorReady (3), or cptEdge (4), in which the Port would operate for this CNPV if cnpdDefModeChoice (32.4.1) had the value cpcAuto (2).

NOTE—cncpAdminDefenseMode (32.3.4) is controlled by the network administrator. The object cnpdAutoDefenseMode (32.4.3) indicates what the CND defense mode would be, if it were controlled by the LLDP Congestion Notification TLV.

32.4.4 cnpdLldpInstanceChoice

Port and priority LLDP instance selector, specifying on which LLDP instance the Congestion Notification TLV is to carry information for this priority on this Port (32.1.1):

- 1) **cnlNone:** No LLDP Congestion Notification TLV is to carry Per-priority CNPV indicators or Per-priority Ready indicators on this Port for this priority;
- 2) **cnlAdmin:** cnpdLldpInstanceSelector (32.4.5) governs which LLDP instance is to carry Per-priority CNPV indicators and Per-priority Ready indicators for this priority in its Congestion Notification TLV on this Port; or
- 3) **cnlComponent:** cncpLldpInstanceSelector (32.3.7) governs LLDP instance selection for this Port and priority (this is the default value).

32.4.5 cnpdLldpInstanceSelector

A reference to the LLDP destination address table entry for an LLDP instance selector, specifying on which LLDP instance the Congestion Notification TLV is to carry information for this priority on this Port, if and only if cnpdLldpInstanceChoice (32.4.4) is set to cnlAdmin (2). Default value is 1. (32.1.1);

32.4.6 cnpdAlternatePriority

(CP only) An alternate priority value to which this priority value is to be remapped when the Port's CND defense mode is cptEdge for this priority, if and only if cnpdDefModeChoice (32.4.1) is set to cpcAdmin (1) (default value 0).

32.4.7 cnpdXmitCnpvCapable

Per-port per-priority Boolean variable indicating whether a given priority on this Port is (TRUE) or is not (FALSE) currently operating as a CNPV. cnpdXmitCnpvCapable is transmitted in the per-priority CNPV indicators in the Congestion Notification TLV (D.2.8.3). The variable is TRUE if and only if all of the following conditions are met:

- a) cngMasterEnable (32.2.1) is TRUE; and
- b) The CND defense mode, as selected by cncpDefModeChoice, cncpAdminDefenseMode, cnpdDefModeChoice, and cnpdAdminDefenseMode, according to Table 32-2, is not cptDisabled.

32.4.8 cnpdXmitReady

Per-port per-priority Boolean variable indicating whether the priority remap defenses for this port and CNPV have been disabled. cnpdXmitReady is transmitted in the per-priority Ready indicators in the Congestion Notification TLV (D.2.8.4). This variable is set and reset by the CND defense state machine (32.6).

32.4.9 cncpDoesEdge

Boolean variable indicating whether the cptEdge CND defense mode (32.1.1) is (TRUE) or is not (FALSE) implemented on this port for this priority. This variable is TRUE in a bridge. In an end station, it may be either TRUE or FALSE. This variable is not a managed object.

NOTE—This variable, when FALSE, causes the state machine in Figure 32-1 to pass through the CNDD_EDGE state directly to the CNDD_INTERIOR state. Since the state machine takes, in theory, no time to execute, it is impossible to tell from any measurement taken outside the system whether or not the actions specified in the CNDD_EDGE state are actually executed.

32.4.10 cnpdAcceptsCnTag

Boolean variable indicating whether the CN-TAG is (TRUE) or is not (FALSE) acceptable on data frames received on this port for this CNPV. This variable is TRUE in a bridge. In an end station, it may be either TRUE or FALSE. This variable is not a managed object.

NOTE—This variable can be used by an end station that finds it convenient for the adjacent bridge to remove CN-TAGs from frames on a CNPV. By preventing this Port and CNPV’s cnpdXmitReady variable from being set TRUE, (see Figure 32-1) cnpdAcceptsCnTag prevents the adjacent device’s CND defense state machine (32.6) from advancing from the CNDD_INTERIOR state to the CNDD_INTERIOR_READY state, and thus prevents it from transmitting a CN-TAG on this CNPV.

32.4.11 cnpdRcvdCnpv

Per-port per-priority Boolean variable indicating the presence of an LLDP neighbor’s Congestion Notification TLV (D.2.8). cnpdRcvdCnpv is TRUE only if all of the following are true:

- a) An LLDP instance is selected by cncpLldpInstanceSelector (32.3.7) and cnpdLldpInstanceSelector (32.4.5);
- b) The selected LLDP instance indicates that the Port has a single neighbor;
- c) The selected LLDP instance’s neighbor information includes a Congestion Notification TLV; and
- d) That Congestion Notification TLV’s Per-priority CNPV indicators field has a value of 1 in the position corresponding to this CNPV.

32.4.12 cnpdRcvdReady

Per-port per-priority Boolean variable indicating that the neighboring system has turned off its CND defenses. cnpdRcvdReady is TRUE only if all of the following are true:

- a) cnpdRcvdCnpv is TRUE; and
- b) The bit in the same Congestion Notification TLV’s (D.2.8) Per-priority Ready indicators field (D.2.8.4) has a value of 1 in the position corresponding to this CNPV.

32.4.13 cnpdIsAdminDefMode

Boolean variable derived from the managed objects, indicating to the Congestion Notification Domain defense state machine (32.6) whether or not the CND defense mode is being forced to a particular state by the managed objects, as shown in Table 32-3.

NOTE—The purpose of this variable is to simplify the Boolean expressions in Figure 32-1.

Table 32-3—Determining cnpdIsAdminDefMode and cnpdDefenseMode

(Per-port per-priority) cnpdDefModeChoice (32.4.1)	(Component per- priority) cnepDefModeChoice (Clause 32)	cnpdIsAdminDefMode (32.4.13) is:	cnpdDefenseMode (32.4.14) is obtained from:
cpcAdmin	Any	TRUE	(Per-port per-priority) cnpdAdminDefenseMode 32.4.2
cpcAuto	Any	FALSE	Not defined
cpcComp	cpcAdmin	TRUE	(Component per-priority) cnepAdminDefenseMode 32.3.4
cpcComp	cpcAuto	FALSE	Not defined

32.4.14 cnpdDefenseMode

Enumerated value, indicating the management choice for the CND defense mode, if any, as shown in Table 32-3. The value of this variable is not defined if the managed variables indicate no choice for the CND defense mode.

NOTE—The purpose of this variable is to simplify the exit condition expressions in Figure 32-1.

32.5 Congestion Notification Domain defense procedures

There are three procedures associated with the Congestion Notification Domain defense state machine. They are as follows:

- a) DisableCnpvRemapping() (32.5.1)
- b) TurnOnCnDefenses() (32.5.2)
- c) TurnOffCnDefenses() (32.5.3)

32.5.1 DisableCnpvRemapping()

Removes any overrides by congestion notification of the Priority Regeneration Table (6.9.4).

32.5.2 TurnOnCnDefenses()

Overrides the Priority Regeneration Table (6.9.4) to use the alternate priority specified by cnepAlternatePriority (32.3.2), cnepAutoAltPri (32.3.3), and cnpdAlternatePriority (32.4.6) for frames with this CNPV, instead of the value in the Priority Regeneration Table.

32.5.3 TurnOffCnDefenses()

Overrides the Priority Regeneration Table (6.9.4) to not alter the priority of incoming frames with this CNPV, instead of using the value in the Priority Regeneration Table.

32.6 Congestion Notification Domain defense state machine

The Congestion Notification Domain defense state machine is illustrated in Figure 32-1. A congestion aware bridge component or end station shall implement one Congestion Notification Domain defense state machine per port per priority level.

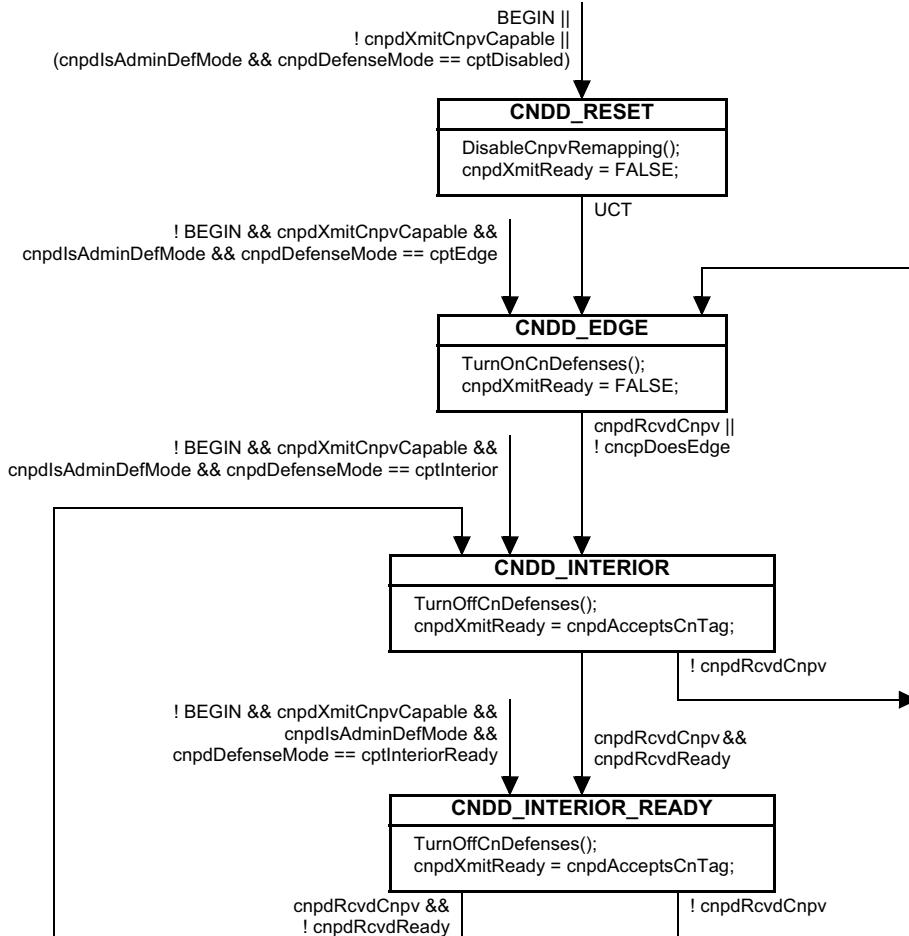


Figure 32-1—Congestion Notification Domain defense state machine

NOTE—In Figure 32-1, global transitions are associated with each value of `cnpdDefenseMode` when the CND defense mode is chosen by the managed variables; the nonglobal transitions from state to state can take place only when the CND defense mode is chosen by LLDP.

32.7 Congestion notification protocol

Clause 32 is a detailed specification of the state machines required to implement the QCN algorithm introduced in 30.2. Table 32-4 shows the correlation between the description of QCN in 30.2, the state machine variables in this clause, and the CNM PDU fields in 33.4.

Table 32-4—Correspondence of QCN protocol and CCF message fields

Quantized Congestion Notification protocol, 30.2	Congestion notification protocol, 32.7	Congestion Notification Message, 33.3
Quantized Feedback calculation (32.8.9, 32.9.4:e)		
F_b (30.2.1)	—	cpFb (33.4.3)
Q_{off} (30.2.1)	— cpQOffset (32.8.7)	cnmQOffset (33.4.5)
Q (30.2.1)	cpQLen (32.8.4)	—
Q_{eq} (30.2.1)	cpQSp (32.8.3)	—
Q_δ (30.2.1)	cpQDelta (32.8.8)	cnmQDelta (33.4.6)
Q_{old} (30.2.1)	cpQLenOld (32.8.5)	—
w (30.2.1)	cpW (32.8.6)	—
RP rate calculations (32.14.5, 32.14.6, Figure 32-2)		
CR (30.2.2)	rpCurrentRate (32.13.6)	—
TR (30.2.2)	rpTargetRate (32.13.5)	—
Byte Counter (30.2.3)	rpByteCount (32.13.2)	—
Timer (30.2.3)	RpWhile (32.12.1)	—
G_d (30.2.2.1)	rpgGd (32.11.8)	—
R_{AI} (30.2.2.1)	rpgAiRate (32.11.6)	—
R_{HAI} (30.2.3)	rpgHaiRate (32.11.7)	—
i (30.2.3)	min(rpByteStage, rpTimeStage) (RPR_HYPER_INCREASE)	—

There are two stateful participants in the Congestion notification protocol. They are as follows:

- a) The Congestion Point (31.1.1) shall behave in a manner that is indistinguishable, from an observer external to a system, from a strict implementation of the variables in 32.8 and the procedures in 32.9.
- b) The Reaction Point (31.2.2.2) shall behave in a manner that is indistinguishable, from an observer external to a system, from a strict implementation of the timer in 32.12, variables in 32.13, the procedures in 32.14, and the state machine in 32.15.

32.8 Congestion Point variables

The following variables control the operation of a CP:

- a) cpMacAddress (32.8.1)
- b) cpId (32.8.2)
- c) cpQSp (32.8.3)
- d) cpQLen (32.8.4)
- e) cpQLenOld (32.8.5)
- f) cpW (32.8.6)
- g) cpQOffset (32.8.7)

- h) cpQDelta (32.8.8)
- i) cpFb (32.8.9)
- j) cpEnqueued (32.8.10)
- k) cpSampleBase (32.8.11)
- l) cpDiscardedFrames (32.8.12)
- m) cpTransmittedFrames (32.8.13)
- n) cpTransmittedCnms (32.8.14)
- o) cpMinHeaderOctets (32.8.15)

32.8.1 cpMacAddress

The MAC address, belonging to the system transmitting the CNM PDU, used as the source _address of Congestion Notification Messages (CNMs) sent from this CP (32.9.4:b).

32.8.2 cpId

Unsigned integer. A number that uniquely identifies a CP in a Virtual Bridged Network (32.9.4:p).

NOTE—This standard does not specify whether the CPID reported in a CNM by a CP that serves multiple CNPVs does or does not have the same value for its different CNPVs.

32.8.3 cpQSp

Unsigned integer. The set-point for the queue. This is the target number of octets in the CP's queue (32.9.4:e). Default value 26000.

32.8.4 cpQLen

Unsigned integer. The current number of octets in the CP's queue (32.9.4:c).

32.8.5 cpQLenOld

Unsigned integer. The previous value of cpQLen. cpQLenOld is updated from cpQLen each time GenerateCnmPdu() is called (32.9.4:c).

32.8.6 cpW

Real number. cpW is the weight to be given to the change in queue length in the calculation of cpFb (32.8.9). Default value 2. Although cpW is specified as a real number, it is constrained to be a power of 2, e.g., 1/2, 1, 2, 4. In practice, therefore, it can be represented as a shift distance (plus an addition) in 32.9.4:e.

32.8.7 cpQOffset

The signed integer value of the transmitting CP's cpQSp (32.8.3) – cpQLen (32.8.4) used to calculate cpFb (32.8.9), and thus the Quantized Feedback field (33.4.3).

32.8.8 cpQDelta

The signed integer value of the transmitting CP's cpQLen (32.8.4) – cpQLenOld (32.8.5) used to calculate cpFb (32.8.9), and thus the Quantized Feedback field (33.4.3).

32.8.9 cpFb

Signed integer. Calculated just before the CP attempts to enqueue a frame:

$$\text{cpFb} = \text{cpQOffset} - \text{cpW} \times \text{cpQDelta},$$

where

$$\text{cpQDelta} = \text{cpQLen} - \text{cpQLenOld}$$

$$\text{cpQOffset} = \text{cpQSp} - \text{cpQLen}$$

cpFb has two terms. The first term is the difference between the current and the desired queue lengths (cpQOffset , 32.8.7). The second is a weight factor cpW times the difference cpQDelta (32.8.8) between the current and the previous queue lengths. Thus, a multiple of the first derivative of the queue size is subtracted from the current nonoptimality of the queue, so that if the queue length is moving toward the set point cpQSp , cpFb will be closer to 0 than if the queue length is moving away from cpQSp .

32.8.10 cpEnqueued

Signed integer. The number of octets remaining to be enqueued by the CP before a CNM PDU is to be generated (32.9.3).

32.8.11 cpSampleBase

Unsigned integer. The minimum number of octets to enqueue in the CP's queue between CNM PDU transmissions (32.9.3). Default value 150 000.

32.8.12 cpDiscardedFrames

The number of frames offered to this CP that were discarded because of a full output queue (31.1.1).

32.8.13 cpTransmittedFrames

The number of data frames enqueued for transmission on this CP's output queue (31.1.1).

32.8.14 cpTransmittedCnms

The number of CNMs transmitted by this CP (32.9.4:s).

32.8.15 cpMinHeaderOctets

The minimum number of octets that the CP is to return in the Encapsulated MSDU field (33.4.10) of each CNM it generates (32.9.4:k). Default value 0.

32.9 Congestion Point procedures

The procedures used in a CP include the following:

- a) Random() (32.9.1)
- b) NewCpSampleBase() (32.9.2)
- c) EM_UNITDATA.request (parameters) (32.9.3)
- d) GenerateCnmPdu() (32.9.4)

A CP needs no state machine; the procedure EM_UNITDATA.request (parameters), called each time a frame is presented for queuing, triggers all of the CP's functions.

32.9.1 Random

The Random function takes two parameters, min and max, and returns a pseudo-random number in the range $\text{min} \leq \text{number} < \text{max}$. This function shall be initialized so that it generates a different value each time the system is reset.

32.9.2 NewCpSampleBase()

Called by EM_UNITDATA.request (parameters) (32.9.3) when generating a new value for cpSampleBase (32.8.11) from the CNM PDU Quantized Feedback field (33.4.3) last generated by GenerateCnmPdu(). NewCpSampleBase() returns a real number according to Table 32-5.

Table 32-5—NewCpSampleBase() return value as a function of cpFb

Quantized Feedback / 8	Value returned by NewCpSampleBase()
0	1
1	1/2
2	1/3
3	1/4
4	1/5
5	1/6
6	1/7
7	1/8

NOTE—The values in Table 32-5 are chosen so that the value returned by NewCpSampleBase() can be an integer to be used in a division, rather than a real number to be used in a multiplication.

32.9.3 EM_UNITDATA.request (parameters)

A CP offers an instance of the EISS (6.8) to the Queuing frames function (8.6.6). When called upon to enqueue a frame, the CP

- a) Determines the number of octets of buffer space required to enqueue the frame, and subtracts that number from cpEnqueued (32.8.10)
- b) If cpEnqueued ≤ 0 , then the CP:
 - 1) Calculates a new value for cpFb according to 32.8.9
 - 2) Calls GenerateCnmPdu() to conditionally transmit a CNM PDU
 - 3) Replaces cpEnqueued with $\text{cpSampleBase} \times \text{NewCpSampleBase}() \times \text{Random}(0.85, 1.15)$, and if the result is less than zero, resets cpEnqueued to 0

The accuracy of the sample jitter introduced by the function Random(0.85, 1.15) is not critical to the success of congestion notification. The amount of jitter introduced by this function should be no more than $0.75 \leq \text{jitter} < 1.25$ and no less than $0.875 \leq \text{jitter} < 1.125$.

32.9.4 GenerateCnmPdu()

Called by the CP to conditionally generate a CN-TAGged CNM PDU for output. GenerateCnmPdu():

- a) Uses the source_address of the frame triggering the CNM PDU as the destination_address of the generated CNM PDU.
- b) Uses cpMacAddress (32.8.1) as the source_address of the CNM PDU.
- c) Replaces the value of cpQLenOld (32.8.5) with cpQLen (32.8.4).
- d) If cpFb is greater than or equal to 0, or if the destination_address parameter of the CNM PDU is a Group, and not an Individual, address, does nothing; no further processing takes place, and no CNM PDU is transmitted.
- e) Fills the Quantized Feedback field (33.4.3) of the CNM PDU as follows:

$$\text{if } (\text{cpFb} < - \text{cpQSp} \times (2 \times \text{cpW} + 1))$$

$$\quad \text{Quantized Feedback} = 63$$

$$\text{else}$$

$$\quad \text{Quantized Feedback} = - \text{cpFb} \times 63 / (\text{cpQSp} \times (2 \times \text{cpW} + 1))$$
- f) If the Quantized Feedback field equal to 0 does nothing; no further processing takes place, and no CNM PDU is transmitted.
- g) If the first octets of the mac_service_data_unit of the frame triggering the CNM PDU are a CN-TAG, copies the Flow Identifier field (33.2.1) from that CN-TAG to the CN-TAG of the CNM, else fills the Flow Identifier field of the CNM's CN-TAG with 0.
- h) Sets the priority parameter of the CNM PDU from cngCnmTransmitPriority (32.2.2).

NOTE—The default value for the priority of a CNM PDU is 6. CNM PDUs are not expected to be sent in high volumes, but undelivered CNM PDUs reduce the ability of congestion notification to prevent lost data frames. Priority 7 is typically reserved for control traffic that, if not delivered, can result in the loss of connectivity in the bridged LAN.

- i) Sets the vlan_identifier of the CNM PDU from the vlan_identifier of the frame triggering the CNM PDU, if the dot1agCfmVlanTable is not implemented, else the Primary VID of that vlan_identifier.
- j) Fills the Encapsulated destination MAC address field (33.4.8) of the CNM PDU with the destination_address parameter of the frame triggering the CNM PDU.
- k) Fills the Encapsulated MSDU field (33.4.10) of the CNM PDU with the first octets of the remainder of the mac_service_data_unit following the CN-TAG, if present, of the frame triggering the CNM PDU. The minimum number of octets is the value of cpMinHeaderOctets (32.8.15), or the whole of the mac_service_data_unit following the CN-TAG, if it is shorter, and the maximum is 64.
- l) Fills the Encapsulated MSDU length field (33.4.9) of the CNM PDU with the number of octets inserted into the Encapsulated MSDU field (33.4.10).
- m) Inserts the encapsulation appropriate to the CP's port, as specified in 33.3, at the beginning of the mac_service_data_unit.

NOTE 3—This encapsulation is changed by the bridge, if required, before the CNM PDU is output on a port that uses a different encapsulation from the CP's port.

- n) Fills the Version field (33.4.1) of the CNM PDU with 0.
- o) Fills the ReservedV field (33.4.2) of the CNM PDU with 0.
- p) Fills the Congestion Point Identifier field (33.4.4) of the CNM PDU from the variable cpId (32.8.2).
- q) Computes and fills the cnmQOffset and cnmQDelta fields (33.4.5, 33.4.6) from the variables cpQOffset and cpQDelta.
- r) Fills the Encapsulated priority field (33.4.7) from the priority parameter of the frame triggering the CNM PDU.
- s) Increments cpTransmittedCnms (32.8.14).

- t) Passes the CN-TAG and CNM PDU as an EM_UNITDATA.indication to the higher layers (Figure 31-1), if the CP is in a bridge, or to the Flow multiplexer (31.2.4), if the CP is in an end station (Figure 31-2).

32.10 Reaction Point per-Port per-CNPV variables

There is one set of these variables per Port per CNPV in a congestion aware end station. A set of Reaction Point per-Port per-CNPV variables is created or deleted whenever a CN Port priority managed object is created or deleted. All of the RPs (if more than one) associated with a given Port and CNPV share a single set of the Reaction Point per-Port per-CNPV variables.

32.10.1 rpppMaxRps

An unsigned integer controlling the maximum number of RPs allowed for this CNPV on this Port (default 1). An end station shall not create more than rpppMaxRps on a given Port, but it can create fewer.

32.10.2 rpppCreatedRps

The number of times an RP's rpEnabled variable has been set TRUE at this priority level on this Port (32.14.4:e).

32.10.3 rpppRpCentiseconds

An integer containing the number of RP-centiseconds accumulated by RPs at this priority level on this Port. This variable is incremented once per centisecond (10 ms) for each RP on this Port whose rpEnabled variable (32.13.1) is TRUE.

NOTE—Reading rpppRpCentiseconds and SysUpTime⁴³ at two different times allows a management entity to compute the average number of RPs present for a priority and Port by dividing the change in rpppRpCentiseconds by the change in SysUpTime (which is measured in centiseconds).

32.11 Reaction Point group variables

There is one set of Reaction Point group variables for each group of RPs belonging to a particular CNPV on a Port in a congestion aware end station. It is an implementation choice whether each RP rate control state machine (32.15) has its own set of Reaction Point group variables or shares them with another RP rate control state machine on the same Port and the same CNPV. This standard does not specify how RPs are grouped together for management purposes, except that all of the RPs controlled by a single set of Reaction Point group variables shall serve a single CNPV. At the extremes, a single set of Reaction Point group variables could control all of the RPs serving the same CNPV, or each RP could be controlled by its own set of Reaction Point group variables. The Reaction Point group variables include the following:

- a) rpgEnable (32.11.1)
- b) rpgTimeReset (32.11.2)
- c) rpgByteReset (32.11.3)
- d) rpgThreshold (32.11.4)
- e) rpgMaxRate (32.11.5)
- f) rpgAiRate (32.11.6)
- g) rpgHaiRate (32.11.7)
- h) rpgGd (32.11.8)
- i) rpgMinDecFac (32.11.9)
- j) rpgMinRate (32.11.10)

⁴³From IETF RFC 3418 (STD 62).

32.11.1 rpgEnable

A Boolean value controlling the rpEnabled variables for all of the RP rate control state machines controlled by this Reaction Point group managed object. If rpgEnable is TRUE, the controlled rpEnabled variables are not held in the FALSE state, thus enabling the RPs to pay attention to received CNMs. If rpgEnable is FALSE, the controlled rpEnabled variables are held in the FALSE state, thus disabling the RPs.

32.11.2 rpgTimeReset

Unsigned integer. RpWhile (32.12.1) is reset to either rpgTimeReset or rpgTimeReset / 2 (with random jitter—see Figure 32-2, RPR_ACTIVE_TIME) each time it reaches 0, according to the state of the RP rate control state machine. Default value 15 ms.

32.11.3 rpgByteReset

Unsigned integer. rpByteCount (32.13.2) is reset to either rpgByteReset or rpgByteReset / 2 (with random jitter—see Figure 32-2, RPR_ACTIVE_BYTE) each time it reaches 0, according to the state of the RP rate control state machine. Default value 150 000 octets.

32.11.4 rpgThreshold

Unsigned integer. Specifies the number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states. Default value is 5.

32.11.5 rpgMaxRate

Unsigned integer. The maximum rate, in bits per second, at which an RP can transmit. Default value is the speed of the Port. A system shall support a nonzero minimum value for rpgMaxRate no larger than 5 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgMaxRate is configurable in multiples of 1 Mbit/s.

32.11.6 rpgAiRate

Unsigned integer. The rate, in bits per second, used to increase rpTargetRate in the RPR_ACTIVE_INCREASE state in Figure 32-2. Default value 5 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgAiRate is configurable in multiples of 1 Mbit/s.

32.11.7 rpgHaiRate

Unsigned integer. The rate, in bits per second, used to increase rpTargetRate in the RPR_HYPER_INCREASE state in Figure 32-2. Default value 50 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgHaiRate is configurable in multiples of 1 Mbit/s.

32.11.8 rpgGd

Real number. Multiplied times the Quantized Feedback field (33.4.3) received in a CNM PDU to decrease rpTargetRate. Default value 1/128. rpgGd is configurable as a power of 2, e.g., 1/256, 1/128, 1/64.

32.11.9 rpgMinDecFac

Real number. The minimum factor by which the current RP transmit rate rpCurrentRate can be changed by reception of a CNM. Default value is 0.5.

32.11.10 rpgMinRate

Unsigned integer. The minimum value, in bits per second, for rpCurrentRate (32.13.6). Default value is 10 Mbit/s. A system shall support a minimum value for rpgMinRate no larger than 10 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgMinDecFac is configurable in multiples of 1 Mbit/s.

32.12 Reaction Point timer

An RP implements one timer as follows:

- a) RpWhile (32.12.1).

32.12.1 RpWhile

Timer counter for controlling the RP rate control state machine (32.15). RpWhile operates normally when rpFreeze (32.13.7) is FALSE, and shall not be decremented when rpFreeze is TRUE.

32.13 Reaction Point variables

Each RP rate control state machine (32.15) has its own set of Reaction Point variables. The Reaction Point variables include the following:

- a) rpEnabled (32.13.1)
- b) rpByteCount (32.13.2)
- c) rpByteStage (32.13.3)
- d) rpTimeStage (32.13.4)
- e) rpTargetRate (32.13.5)
- f) rpCurrentRate (32.13.6)
- g) rpFreeze (32.13.7)
- h) rpLimiterRate (32.13.8)
- i) rpFb (32.13.9)

32.13.1 rpEnabled

Boolean. Controls RP rate control state machine. If TRUE, state machine is running. If FALSE, state machine is held in the RPR_RESET state. Initialized to FALSE when an RP rate control state machine is created. Held in the FALSE state if rpgEnable (32.11.1) has the value FALSE.

32.13.2 rpByteCount

Unsigned integer. Counts octets passed from the Rate Limiter (31.2.2.4) to the Flow Selection function (31.2.2.5). The counting down of rpByteCount to or past 0 triggers a reevaluation of rpCurrentRate (32.13.6).

32.13.3 rpByteStage

Unsigned integer. Counts the number of times rpByteCount has counted down to or past 0 since the last CNM was received.

32.13.4 rpTimeStage

Unsigned integer. Counts the number of times RpWhile has counted down to 0 since the last CNM was received.

32.13.5 rpTargetRate

Unsigned integer. The target rate, in bits per second, toward which rpCurrentRate is heading. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc.

32.13.6 rpCurrentRate

Unsigned integer. The rate, in bits per second, at which the Rate Limiter (31.2.2.4) transmits data from its Flow queue when rpFreeze (32.13.7) is FALSE. (The actual output of the Rate Limiter is controlled by rpLimiterRate, 32.13.8.) This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc.

32.13.7 rpFreeze

Boolean. When FALSE, the RP and Rate Limiter operate normally. When TRUE, rpLimiterRate is held to the value 0, to suppress the transmission of frames. rpFreeze is FALSE when there is room for frames from this RP in the output queue, and TRUE when not.

32.13.8 rpLimiterRate

Unsigned integer. The rate, in bits per second, at which the Rate Limiter (31.2.2.4) transmits data from its Flow queue. rpLimiterRate is equal to rpCurrentRate (32.13.6) as long as rpFreeze (32.13.7) is FALSE, and 0, when rpFreeze is TRUE. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc.

32.13.9 rpFb

Unsigned integer. The value last received in the Quantized Feedback field (33.4.3) of a CNM PDU or 0.

32.14 Reaction Point procedures

The following procedures are used by the RP rate control state machine and the Output flow segregation function (31.2.1):

- a) ResetCnm (32.14.1)
- b) TestRpTerminate (32.14.2)
- c) TransmitDataFrame (32.14.3)
- d) ReceiveCnm (32.14.4)
- e) ProcessCnm (32.14.5)
- f) AdjustRates (32.14.6)

32.14.1 ResetCnm

Called whenever the RP rate control state machine enters the RPR_RESET state (Figure 32-2) to set the following variables:

- rpCurrentRate = rpgMaxRate
- rpTargetRate = rpgMaxRate
- rpByteCount = rpgByteReset
- rpFb = 0

32.14.2 TestRpTerminate

Called by the Flow Selection function (31.2.2.5) each time a frame is passed from the Per-CNPV station function's Rate Limiter (31.2.2.4) to the Flow multiplexer (31.2.4). If

- a) rpCurrentRate equals rpgMaxRate
- b) The Flow queue is empty
- c) rpEnabled is TRUE

Then TestRpTerminate sets rpEnabled (32.13.1) to FALSE, which disables the RP until another CNM is received.

NOTE—Condition b) can be met when all frames limited by a given Rate Limiter have been processed. See 31.2.2.1.

32.14.3 TransmitDataFrame

TransmitDataFrame() is called each time a frame is passed from a Rate Limiter (31.2.2.4) to the Port's Flow multiplexer (31.2.4). TransmitDataFrame() subtracts the length of the transmitted frame from rpByteCount (32.13.2) and may insert a CN-TAG containing the Flow ID of the Flow queue at the beginning of the frame's mac_service_data_unit.

If an end station is VLAN-aware or priority-aware, and if a CN-TAG is added to a frame, the CN-TAG shall follow the VLAN or priority tag in the frame.

32.14.4 ReceiveCnm

Called whenever a CNM is received. ReceiveCnm() performs the following actions:

- a) The CNM is validated according to 33.4.11 and is discarded if invalid.
- b) If the CNM frame carries a CN-TAG (33.2), the Flow Identifier from that CN-TAG (33.2.1) is used to identify the particular RP rate control state machine to which this CNM applies.
- c) If the receiving end station is unable to identify the particular RP rate control state machine to which this CNM applies, it discards the CNM, and no further processing takes place.
- d) Sets the rpFb variable from the CNM's Quantized Feedback field (33.4.3).
- e) If the selected RP rate control state machine's rpEnabled variable is FALSE and is not held FALSE because rpgEnable (32.11.1) has the value FALSE, and the CNM's cnmQOffset field (33.4.5) is negative, then rpEnabled is reset to TRUE, the variable rpppCreatedRps (32.10.2) is incremented.

These actions activate the selected RP rate control state machine, if it was not active ($\text{rpEnabled} == \text{FALSE}$), and also cause the RP rate control state machine to process the received CNM.

32.14.5 ProcessCnm

Called whenever the RP rate control state machine enters the RPR_CNM_RECEIVED state (Figure 32-2) to perform the following actions:

```

if (rpByteStage != 0) {
    rpTargetRate = rpCurrentRate;
    rpByteCount = rpgByteReset;
}
rpByteStage = 0;
rpTimeStage = 0;
dec_factor = (1 - (rpgGd * rpFb));
if (dec_factor < rpgMinDecFac)
    dec_factor = rpgMinDecFac;
rpCurrentRate = rpCurrentRate * dec_factor;
if (rpCurrentRate < rpgMinRate)
    rpCurrentRate = rpgMinRate;
RpWhile = rpgTimeReset;

```

32.14.6 AdjustRates

AdjustRates is called whenever the RPR_ADJUST_RATES, RPR_ACTIVE_INCREASE, or RPR_HYPER_INCREASE states are entered. It takes one parameter, *increase*, that specifies how much to increase rpTargetRate, and performs the following actions:

```

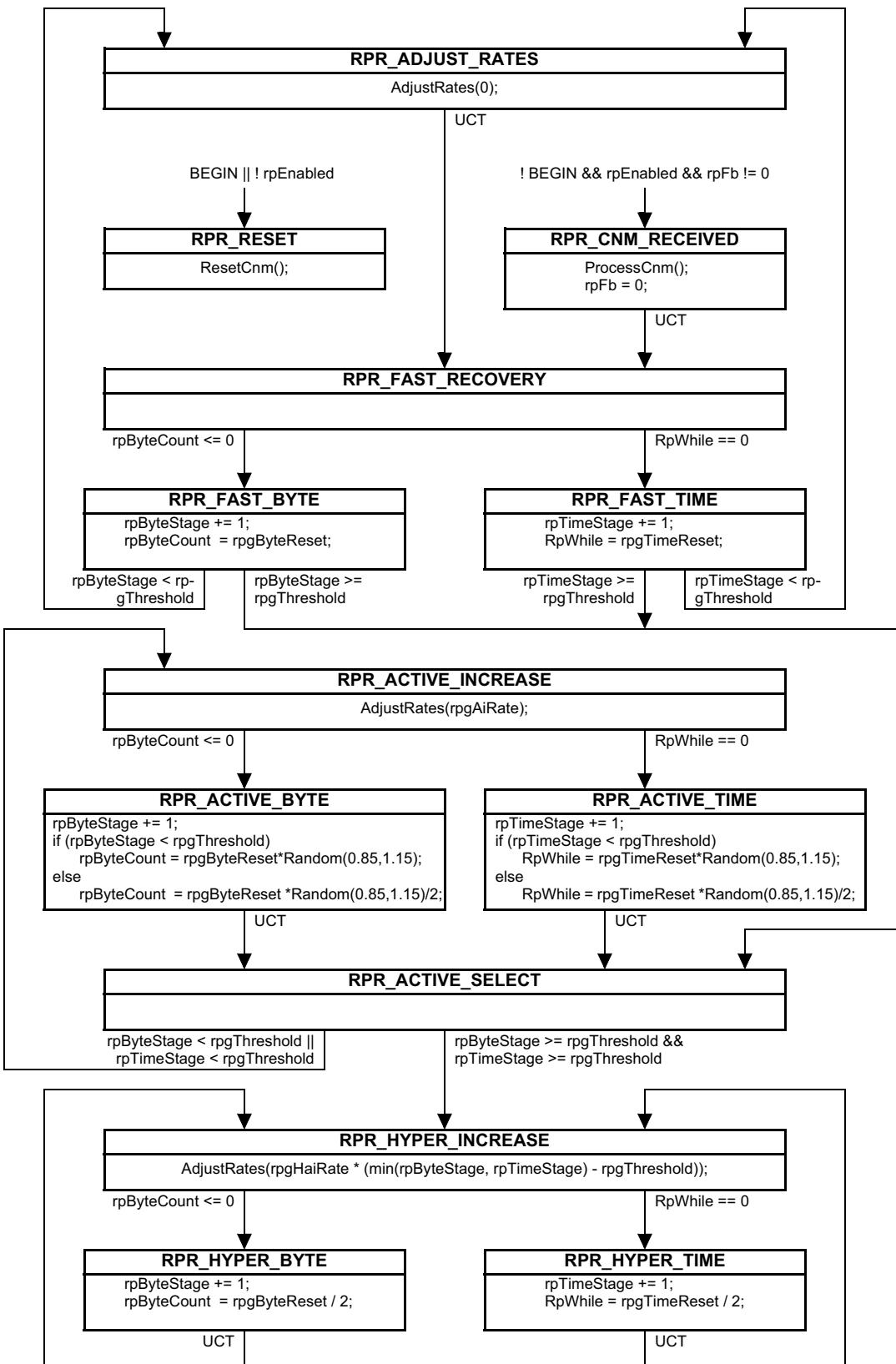
if ((rpByteStage == 1) || (rpTimeStage == 1)) && (rpTargetRate > 10 * rpCurrentRate))
    rpTargetRate = rpTargetRate / 8;
else
    rpTargetRate = rpTargetRate + increase;
rpCurrentRate = (rpTargetRate + rpCurrentRate)/2;
if (rpCurrentRate > rpgMaxRate)
    rpCurrentRate = rpgMaxRate;

```

32.15 RP rate control state machine

The RP rate control state machine is illustrated in Figure 32-2.

In this figure, the RPR_RESET state has no exit. The state machine is held in the RPR_RESET state as long as either the general system initialization variable “BEGIN” is TRUE or the rpEnabled variable, local to the particular instance of the RP rate control state machine, is FALSE. When “BEGIN” becomes FALSE and rpEnabled TRUE, the state machine remains in the RPR_RESET state until a CNM is received by ReceiveCnm (32.14.4) with a Quantized Feedback field (33.4.3) that is less than 0. ReceiveCnm places that field into rpFb (32.13.9), which condition forces the state machine into the RPR_CNM_RECEIVED state. When that state resets rpFb to 0, progress through the state machine continues either until another CNM is received and rpFb is set to a nonzero value, or until the state machine is deactivated by TestRpTerminate (32.14.2) setting rpEnabled to FALSE.

**Figure 32-2—RP rate control state machine**

32.16 Congestion notification and encapsulation interworking function

As mentioned in 30.8, a hierarchical Bridged Network, e.g., a Provider Backbone Bridged Network, can require a congestion notification and encapsulation interworking function for a CP in the core of the network to generate a CNM that can reach the RP, which is presumably outside the core. This general case is illustrated in Figure 32-3. In the network illustrated, encapsulation and decapsulation functions peer with each other to encapsulate the data frames inside some kind of wrapper, e.g., an S-TAG and/or an I-TAG.

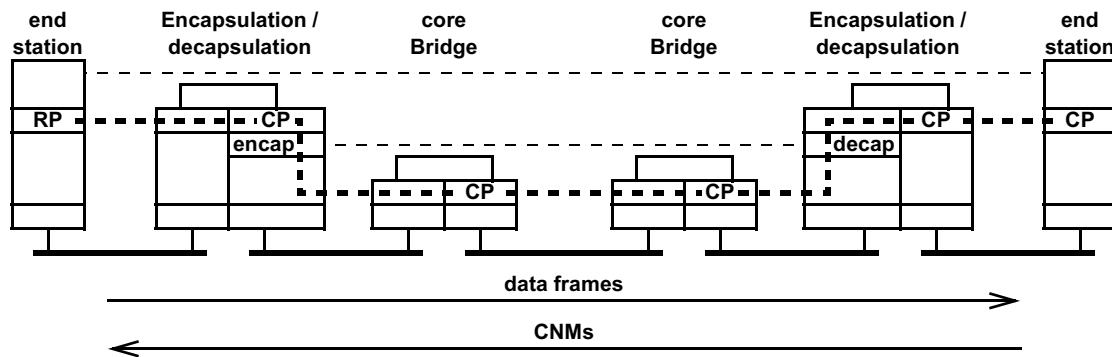


Figure 32-3—CP–RP peering in any hierarchical Bridged Network

Not all encapsulation schemes are equivalent. It is useful to divide hierarchical Bridged Networks into two categories for the purpose of defining congestion notification and encapsulation interworking functions, based on the `mac_service_data_unit` of a data frame carrying (optionally) a CN-TAG as seen by a CP residing in a core Bridge, with one of these cases further subdivided as follows:

- In a network where the original end station's `mac_service_data_unit`, including the optional CN-TAG, comprises the first octets of the `mac_service_data_unit` at the core Bridge's CP, there is no need for a congestion notification and encapsulation interworking function. For example, a Provider Bridged network in which the C-TAGs are not carried across the core needs no interworking function.
- In a network where an encapsulation comprises the first octets of the `mac_service_data_unit` at the core Bridge's CP, a Congestion notification and encapsulation interworking function is required. It is useful to make a further distinction, in this case:
 - The MAC addresses of Ports in the Bridges where the encapsulation and decapsulation functions are performed are the destination and source addresses of frames crossing the core Bridges. For example, in a Provider Backbone Bridged Network, the I-TAG, and perhaps an S-TAG and/or a C-TAG as well, can be present as the first octets of the `mac_service_data_unit`, and the `source_address` and `destination_address` parameters visible to the core Bridge's CP are the addresses of PIPs in Backbone Edge Bridges.
 - In other networks, although an encapsulation is in the first octets of the `mac_service_data_unit` at the core Bridge's CP, the source and destination MAC addresses are not altered by the encapsulation and decapsulation operations. For example, in a Provider Bridged network in which both a C-TAG and an S-TAG are carried across the core, the C-TAG occupies the first octets of the `mac_service_data_unit` as seen by a core Bridge's CP.

In either type of network covered by case b, the core Bridge's CP is configured by the network administrator to return, in the Encapsulated MSDU field of the CNM PDU (33.4.10), the leading octets of the `mac_service_data_unit` of the data frame triggering transmission of the CNM. Since the CN-TAG of the data frame, if present, is buried behind encapsulations unknown to the CP, when the CN-TAG and CNM are built according to 32.9.4, the Flow Identifier returned in the CN-TAG of the CNM is filled with 0. The network

administrator configures the CP to return at least enough octets of the data frame's `mac_service_data_unit` to include the CN-TAG of the original data frame transmitted by the RP.

The difference between case 1 and case 2, preceding, lies in the means used by the Port containing the Congestion notification and encapsulation interworking function to recognize the CNM in order to translate it. In case 1, the CNM is addressed to that Port, and has a CN-TAG and the CNM encapsulation, instead of a data frame tag and encapsulation. In case 2, the only reliable indicator of a CNM that needs to be transformed, as opposed for example, to a CNM returned by the destination end station's CP, can be the presence of an encapsulation known to the interworking function in the Encapsulated MSDU field (33.4.10) of the CNM PDU.

When the CNM generated by the core Bridge's CP reaches a congestion notification and encapsulation interworking function, that interworking function uses the information in the CNM PDU, including the encapsulated data frame information, to construct a CN-TAG and CNM PDU to send to the RP. The steps taken are as follows:

- c) In case 1, where the CNM is returned to the MAC address of the interworking function's Port, the `destination_address` parameter is obtained from the appropriate source address found in the Encapsulated MSDU field, and the `source_address` parameter is a MAC address belonging to the Bridge containing the interworking function, and valid in the context of the newly generated CNM, for example, a management port address.
- d) In case 2, where the CNM is recognized by the presence of the encapsulation in the Encapsulated MSDU field, the `source_address`, and `destination_address` parameters are unchanged.
- e) The `vlan_identifier` is obtained from the Encapsulated MSDU field.
- f) The `drop_eligible` parameter is False.
- g) The `priority` parameter used is the value configured for a CP on a Port in the Bridge in which the interworking function resides, and through which the CNM passes.
- h) The Encapsulated priority field (33.4.7) is obtained from the Encapsulated MSDU field, or if not present there, the Encapsulated priority is left unchanged.

NOTE—In spite of the provisions for CND defense in this clause, it is possible for a network administrator to configure a network so that the priority of a CCF data frame is not preserved. For example, in a Provider Bridged Network, two CNPVs could be mapped to the same S-TAG priority, and the C-TAG not configured to be carried across the network. In that case, an end station with an RP on each of those CNPVs, and that does not distinguish between those RPs using CN-TAGs, would be unable to assign returned CNMs to the right RP. At least some of its CCFs would not be constrained, and uncontrolled congestion would likely result. It is therefore recommended that the CN-TAG be included across a Provider Bridged Network, or that each CNPV be mapped to a separate priority in the backbone.

- i) If a CN-TAG is found in the Encapsulated MSDU, its Flow Identifier is copied to the CN-TAG of the newly generated CNM, else 0 is used for that CN-TAG's Flow Identifier.
- j) The encapsulation in the Encapsulated MSDU field of the CNM PDU, up to and including the CN-TAG (which is not necessarily present), is elided, the remainder (if any) of the Encapsulated MSDU field moved up to the first octets of the field, and the Encapsulated MSDU length field reduced accordingly.

33. Encoding of congestion notification Protocol Data Units

This clause specifies the method of encoding congestion notification PDUs. The specifications include the following:

- a) The general structure and encoding of congestion notification PDUs (33.1).
- b) The format used for the Congestion Notification Tag (CN-TAG, 33.2).
- c) The encapsulation used for a Congestion Notification Message (CNM, 33.3).
- d) The format used for the CNM (33.4).

The format of the IEEE Std 802.1AB-2009 Type Length Value (TLV) used to signal that a VLAN-aware Bridge or end station is congestion aware, and the state of its Congestion Notification Domain defense mode, is specified in D.2.8.

NOTE—Clause 30 introduces the principles of congestion notification operation and the network architectural concepts that support it. Clause 31 breaks down the congestion notification entities into their components. Clause 32 specifies the protocols operated by these components.

33.1 Structure, representation, and encoding

All congestion notification PDUs contain an integral number of octets.

The octets in a congestion notification 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 congestion notification entity.

The bits in an octet are numbered from 1 to 8 in order of increasing bit significance, where 1 is the LSB in the octet.

Where octets and bits within a congestion notification 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.

33.2 Congestion Notification Tag format

The means for identifying Congestion Notification Tag PDUs depend on the medium. For media using a Length/Type field, e.g., IEEE 802.3 media, the identification consists of two octets containing the EtherType value shown (in hexadecimal notation) in Table 33-3. Media requiring an LLC encapsulation (e.g., IEEE 802.11) use the SNAP encoding shown (in hexadecimal notation) in Table 33-4.

Table 33-1—Congestion Notification Tag Encapsulation: Length/Type Media

Octet	Length
Tag EtherType (0x22E9)	1 2
Flow Identifier	3 2

Table 33-2—Congestion Notification Tag Encapsulation: LLC Media

Octet	Length
0xAAAA03	1 3
0x000000	4 3
Tag EtherType (0x22E9)	7 2
Flow Identifier	9 2

In a VLAN-aware Bridge, the shim that supports the Enhanced Internal Sublayer Service (6.1, 6.8, also see Figure 22-4) is below the Queuing shim (8.6.6, 8.6.7, 8.6.8, 31.1.1). The CN-TAG is examined by the CP, which is part of the Queuing shim. Therefore, the CN-TAG will be further from the source_address and destination_address fields, and closer to the end of the frame, than a VLAN tag added by 6.8, if any. In order for a bridge to be able to recognize the CN-TAG in a data frame, an end station shall insert the CN-TAG and VLAN tag in the relative order required by a bridge: addresses, VLAN tag, CN-TAG, data.⁴⁴

33.2.1 Flow Identifier

The meaning and encoding of the Flow Identifier field are not specified by this standard. A CP shall not interpret the value in a Flow Identifier field. The Flow Identifier returned in the CN-TAG of a CNM is used by an end station's CNM demultiplexer (31.2.5) to identify the RP, the Flow queue, or a group of one or more individual flows, from which was transmitted the data frame that triggered the CNM.

33.3 Congestion Notification Message

The means for identifying Congestion Notification Message PDUs depend on the medium. For media using a Length/Type field, e.g., IEEE 802.3 media, the identification consists of two octets containing the EtherType value shown (in hexadecimal notation) in Table 33-3. Media requiring an LLC encapsulation (e.g., IEEE 802.11) use the SNAP encoding shown (in hexadecimal notation) in Table 33-4.

Table 33-3—Congestion Notification Message Encapsulation: Length/Type Media

Octet	Length
PDU EtherType (0x22E7)	1 2
CNM PDU	3 24 to 88

⁴⁴There are other tags, e.g., the IEEE 802.1AE MAC security tag, not included in this abbreviated list.

Table 33-4—Congestion Notification Message Encapsulation: LLC Media

Octet	Length
0xAAAA03	1
0x000000	4
PDU EtherType (0x22E7)	7
CNM PDU	9 24 to 88

33.4 Congestion Notification Message PDU format

The format of a Congestion Notification Message PDU is illustrated in Table 33-5. If the frame that triggered the transmission of the CNM carried a CN-TAG, then that CN-TAG is retained in the frame carrying the CNM PDU; otherwise, a CN-TAG with a FlowID of 0 is used in the frame carrying the CNM PDU.

Table 33-5—Congestion Notification Message PDU

Octet	Length
Version	1 4 bits
ReservedV	1, 2 6 bits
Quantized Feedback	2 6 bits
Congestion Point Identifier (CPIID)	3 8
cnmQOffset	11 2
cnmQDelta	13 2
Encapsulated priority	15 2
Encapsulated destination MAC address	17 6
Encapsulated MSDU length	23 2
Encapsulated MSDU	25 0–64

33.4.1 Version

This field, 4 bits in length, shall be transmitted with the value 0, and shall be ignored on receipt. The Version field occupies the most significant bits of the first octet of the CNM PDU.

33.4.2 ReservedV

This field, 6 bits in length, shall be transmitted with the value 0, and shall be ignored on receipt. The ReservedV field occupies the least significant 4 bits of the first octet, and the most significant 2 bits of the second octet, of the CNM PDU.

33.4.3 Quantized Feedback

This field, 6 bits in length, contains the Quantized Feedback value calculated by the CP's EM_UNITDATA.request (parameters) procedure (32.9.3). The Quantized Feedback field occupies the least significant bits of the first two octets of the CNM PDU.

33.4.4 Congestion Point Identifier

This field, 8 octets in length, uniquely identifies the CP that triggered the transmission of this Congestion Notification Message PDU format within the Congestion Notification Domain. Because the transmitting system can format the CPID in any manner, a receiving RP shall not assign any meaning to subfields within a CPID.

NOTE—Making the CPID 8 octets long removes the connection between CP identity and the source MAC address of the CNM PDU. One way to achieve the requirements for uniqueness of the CPID would be to use a Port's MAC address as the high-order 6 octets of the CPID, and use the low-order two octets of the CPID to hold the priority of the CP's queue and/or an indication of the physical port.

33.4.5 cnmQOffset

The two's-complement signed integer value of the transmitting CP's cpQOffset (32.8.7) in units of 64 octets. This standard does not specify whether cpQOffset is rounded or truncated to obtain cnmQOffset. If the value of cpQOffset is less than -32768, a value of -32768 is used in cpQOffset, and if greater than 32767, 32767 is used.

33.4.6 cnmQDelta

The two's-complement signed integer value of the transmitting CP's cpQDelta (32.8.8) in units of 64 octets. This standard does not specify whether cpQDelta is rounded or truncated to obtain cnmQDelta. If the value of cpQDelta is less than -32768, a value of -32768 is used in cnmQDelta, and if greater than 32767, 32767 is used.

NOTE—Although cnmQDelta is not used by the RP, it is included in the CNM PDU to aid the network administrator when adjusting configuration parameters.

33.4.7 Encapsulated priority

This field, 2 octets in length, contains the priority parameter of the frame that triggered the transmission of this Congestion Notification Message in the most significant 3 bits of the field. The remaining bits of the field are transmitted as 0, and ignored on receipt.

33.4.8 Encapsulated destination MAC address

This field, 6 octets in length, contains the destination_mac_address parameter of the frame that triggered the transmission of this Congestion Notification Message.

33.4.9 Encapsulated MSDU length

This field, 2 octets in length, contains the number of octets returned in the Encapsulated MSDU field (33.4.10).

33.4.10 Encapsulated MSDU

This field, a maximum of 64 octets in length, contains the initial octets of the mac_service_data_unit of the frame that triggered the transmission of this Congestion Notification Message.

33.4.11 CNM Validation

A CNM PDU received by a CNM demultiplexer (31.2.5) shall be considered invalid and be discarded if the following condition is TRUE:

- a) There are fewer than 24 octets in the mac_service_data_unit.

A CNM PDU received by a CNM demultiplexer (31.2.5) may be considered invalid and be discarded if the following condition is TRUE:

- b) The CNM PDU has no CN-TAG.

NOTE 1—The variable cnpdAcceptsCnTag (32.4.10), when set FALSE in an end station, indicates to the neighboring Bridge that the CN-TAG is to be removed from all frames, including CNMs, transmitted to the end station. It is not recommended that an end station set cnpdAcceptsCnTag to FALSE if this would result in the end station discarding CNMs for lack of a CN-TAG.

The following condition shall not cause a received CNM PDU to be considered invalid:

- c) There are nonzero bits in the Version (33.4.1) or ReservedV (33.4.2).

NOTE 2—These rules permit information to be added to the CNM PDU following the Encapsulated MSDU field (33.4.10) in later revisions of this standard.

34. Forwarding and queuing for time-sensitive streams

34.1 Overview

This clause describes a set of tools that can be used to support the forwarding and queuing requirements of time-sensitive streams. In this context, a “time-sensitive stream” is taken to be a stream of traffic, transmitted from a single source station, destined for one or more destination stations, where the traffic is sensitive to timely delivery, and in particular, requires transmission latency to be bounded. Such streams include video or audio data streams, where there is a desire to limit the amount of buffering required in the receiving station.

NOTE 1—An example of this requirement would be a live performance where a video image of the performance is simultaneously projected on a screen in the auditorium, and it is desirable for the sound and image to be “in synch” with the performance.

In order to address the needs of such traffic in Bridges, the following are provided:

- a) A means of detecting the boundary between a set of Bridges that support SRP (an SRP domain) and surrounding Bridges that do not support SRP. This mechanism is described in 34.2.

NOTE 2—The primary intent of these functions is to support SRP; however, there is no specific interdependency between these functions and SRP, so they could equally be used to support other admission control mechanisms if they were implemented.

- b) A set of bandwidth availability parameters for each port that are used to record the maximum bandwidth available to a given outbound queue, and the actual bandwidth reserved, for that queue. These parameters are described in 34.3.
- c) A credit-based shaper algorithm, defined in 8.6.8.1, that is used to shape the transmission of stream-based traffic in accordance with the bandwidth that has been reserved on a given outbound queue.
- d) A discussion of the relationship between the size of the layer 2 “payload” (the MSDU) carried in a frame and how that relates to the actual bandwidth that will be consumed when that MSDU is transmitted on a particular Port (34.4).
- e) An algorithm for determining the mapping of the priorities associated with received frames onto the traffic classes available on the transmission Ports of a Bridge (34.5).
- f) A definition of the required behavior of an end station that acts as the source of a time-sensitive data stream (34.6).

34.2 Detection of SRP domains

The concept of AV streams, the Stream Reservation Protocol (SRP), and the traffic forwarding and shaping functions that support stream transmission (see 6.9.4 and 8.6.8.1), rely on the ability of each bridge to detect whether each of its ports is at the edge of a region of connected bridges that support SRP on a particular priority, so that the priority code point (PCP) values associated with traffic entering an *SRP domain* (3.176) can be properly regenerated at the boundary of the domain, as described in 6.9.4.

Bridges detect the edge of an SRP domain by observing SRP behavior. If a Bridge receives SRP registrations using a particular priority, then it is reasonable to believe that they are being received from an SRP capable device; the SRP engine can therefore signal which Ports of a Bridge are at the boundary of an SRP domain. The per-port, per-SR class, *SRPdomainBoundaryPort* parameter determines whether or not a Port is considered to be at the edge of an SRP domain or within the core of the domain, as defined in 6.6.4. This parameter is controlled by the operation of SRP.

NOTE 1—SRP domain detection is based on the assumption that a device connected to a Port either is SRP capable for a given SR class, or is not SRP capable for that SR class. SRP provides a boundary detection mechanism through the

exchange of MSRPDUs; the boundary of a domain therefore expands to include Ports as SRP attributes are declared. The position of the domain boundary has no effect on the transmission of SRP frames; rather, it reflects where SRP activity is occurring. Ports are removed from the SRP domain when they are removed from the active topology.

34.3 The bandwidth availability parameters

The following bandwidth availability parameters exist for each Port, and for each traffic class, N, that supports the credit-based shaper algorithm:

- a) *portTransmitRate* as defined in 8.6.8.2;
- b) *deltaBandwidth(N)*: The additional bandwidth, represented as a percentage of *portTransmitRate*, that can be reserved for use by the queue associated with traffic class N, in addition to the *deltaBandwidth(N)* values associated with higher priority queues. For a given traffic class N, the total bandwidth that can be reserved is the sum of the *deltaBandwidth* values for traffic class N and all higher traffic classes, minus any bandwidth reserved by higher traffic classes that support the credit-based shaper algorithm (see 34.3.1).
- c) *adminIdleSlope(N)*: The bandwidth, in bits per second, that has been requested by management to be reserved for use by the queue associated with traffic class N. If SRP is in operation, this parameter has no effect; if SRP is not in operation, then the value of *operIdleSlope(N)* is always equal to the value of *adminIdleSlope(N)*.
- d) *operIdleSlope(N)*: The actual bandwidth, in bits per second, that is currently reserved for use by the queue associated with traffic class N. This value is used by the credit-based shaper algorithm (8.6.8.2) as the *idleSlope* for the corresponding queue.

In all cases, bandwidth is defined in terms of the actual bandwidth consumed when frames are transmitted through the Port, not the size of the MSDU “payload” carried within those frames. 34.4 discusses the relationship between MSDU size and actual bandwidth consumed.

NOTE—While the *deltaBandwidth* values are specified with respect to specific traffic classes, and indicate the amount of bandwidth that can be reserved for traffic belonging to a particular traffic class, this does not imply that these traffic classes have preferential access to that portion of the bandwidth. The priority of a given traffic class does not, for example, imply anything about the importance of a data stream that uses that class; the reservation strategy might therefore allocate bandwidth to a high importance stream that uses a lower priority traffic class in preference to a stream of lower importance that uses a higher priority traffic class.

34.3.1 Relationships among bandwidth availability parameters

The recommended default value of *deltaBandwidth(N)* for the highest numbered traffic class supported is 75%, and for any lower numbered traffic classes, the recommended default value is 0%. The *deltaBandwidth(N)* for a given N, plus the *deltaBandwidth(N)* values for any higher priority queues (larger values of N) defines the total percentage of the Port’s bandwidth that can be reserved for that queue and all higher priority queues. For the highest priority queue, this means that the maximum value of *operIdleSlope(N)* is *deltaBandwidth(N)%* of *portTransmitRate*. However, if *operIdleSlope(N)* is actually less than this maximum value, any lower priority queue that supports the credit-based shaper algorithm can make use of the reservable bandwidth that is unused by the higher priority queue. So, for queue N-1, the maximum value of (*operIdleSlope(N)* + *operIdleSlope(N-1)*) is (*deltaBandwidth(N)* + *deltaBandwidth(N-1)*)% of *portTransmitRate*.

NOTE 1—For example, suppose two queues, 3 and 2, support the credit-based shaper algorithm for SR classes A and B, respectively. Suppose *deltaBandwidth(3)* for SR class A is currently 20%, and *deltaBandwidth(2)* for SR class B is currently 30%. If *operIdleSlope(3)* is currently 10% of *portTransmitRate*, then half of queue 3’s maximum allocation is unused, and the maximum value of *operIdleSlope(2)* is therefore currently 40% of *portTransmitRate*. However, if *operIdleSlope(3)* increases to the full 20% that it is entitled to use, the maximum value of *operIdleSlope(2)* reduces to 30% of *portTransmitRate*.

NOTE 2—The sum of the $\delta\text{Bandwidth}(N)$ values for all values of N should be chosen such that there is sufficient bandwidth available for any nonreserved (best-effort, strict-priority) traffic; the default values are chosen such that the sum of the $\delta\text{Bandwidth}(N)$ values is 75%, so no more than 75% of the Port's available bandwidth is permitted to be reserved. This ensures that when using default settings, there is at least 25% of the Port's bandwidth available for nonreserved traffic. However, as these default settings may be inappropriate for some situations (e.g., links that offer very high bandwidth, or networks with very low levels of nonreserved traffic), they can be modified by management.

34.3.2 Bandwidth availability parameter management

The values of $\delta\text{Bandwidth}(N)$ and $\text{adminIdleSlope}(N)$ can be changed by management, using the management operations defined in 12.20. If the stream reservation mechanisms defined in SRP are supported, then the values of $\text{operIdleSlope}(N)$ are determined solely by the operation of SRP. If the stream reservation mechanisms defined in SRP are not supported, then the values of $\text{operIdleSlope}(N)$ are equal to the values requested by management in the corresponding $\text{adminIdleSlope}(N)$ parameters.

It is possible for the value of portTransmitRate for a Port to change as a result of the normal operation of the underlying MAC service (e.g., as a result of the operation of IEEE 802.1AX link aggregation, or as a result of dynamic changes in bandwidth of the physical layer technology itself, as can occur in wireless LAN technologies); it is also possible for management action to change the values of $\delta\text{Bandwidth}(N)$ for a Port. In either case, the consequence could be one of the following:

- a) The sum of the $\text{operIdleSlope}(N)$ values for the Port could now exceed the total reservable bandwidth allowed for the Port, or the $\text{operIdleSlope}(N)$ value for a given queue could now exceed the reservable bandwidth allowed for the queue, as defined in 34.3.1. Consequently, there could be streams currently active on the Port that can no longer be supported.
- b) The bandwidth now available to a given queue could mean that there are streams that are currently inactive that could be supported on the Port.
- c) Active streams that continue to be supported after the change could see their latency guarantee change.

In either case, corrective action, either by management or by the stream reservation mechanisms defined in SRP, is required in order to restore the parameters to a consistent set of values. In order to indicate that there have been changes in the bandwidth availability database, a signal, $\text{bandwidthAvailabilityChanged}$, is generated for each Port whenever the bandwidth availability parameters portTransmitRate or $\delta\text{Bandwidth}(N)$ are changed; this signal is used by SRP to trigger recalculation of the set of streams that can be reserved on the Port.

NOTE—During recalculation of stream reservations, the Bridge might be temporarily unable to honor bandwidth reservation commitments or forward best-effort traffic.

34.4 Deriving actual bandwidth requirements from the size of the MSDU

The forwarding and queuing mechanisms defined in this clause use bandwidth parameters that are defined in terms of the actual bandwidth used when frames are transmitted on the medium that supports the MAC service available through the Port. In contrast, the Stream Reservation Protocol (SRP) makes use of a traffic specification (TSpec) for each stream that defines the maximum number of bits per frame (MaxFrameSize), of the $\text{mac_service_data_unit}$ parameter that is relayed by the relay function of the Bridge, and a maximum frame rate (MaxIntervalFrames), in frames per class measurement interval, for that stream; i.e., the TSpec takes no account of the per-frame overhead associated with transmitting the MSDU over a given medium. However, when SRP determines the value to be used for the $\text{operIdleSlope}(N)$ parameter associated with a given queue, it is necessary for this value to include the per-frame overhead that will be incurred when frames are transmitted on that Port.

NOTE 1—The frame rate in a TSpec is measured over a “class measurement interval” that depends upon the SR class associated with the stream. SR class A corresponds to a class measurement interval of 125 µs; SR class B corresponds to

a class measurement interval of 250 μ s. These class measurement intervals apply at the source of the stream, i.e., the “talker” end station, and do not necessarily hold good for subsequent stages in the stream’s transmission across a Bridged LAN.

For the purposes of calculating the bandwidth consumption of a stream, it is assumed that the stream data is essentially of constant size and transmission rate, so these maxima can be used to directly define an assumed maximum payload size and the maximum frame rate in frames per second; i.e.,:

$$\text{assumedPayloadSize} = \text{MaxFrameSize} \quad (1)$$

$$\text{maxFrameRate} = \text{MaxIntervalFrames} \times (1/\text{classMeasurementInterval}) \quad (2)$$

where *classMeasurementInterval* is measured in seconds.

NOTE 2—As stated, the calculation of bandwidth from TSpec parameters assumes that the stream data is essentially of constant frame size, and hence, the approximations shown in this section are valid. If the data varies significantly in frame size, then the calculation of per-frame overhead using these assumptions could be significantly in error.

From this, and also from local knowledge of the protocol stack that supports the Bridge Port, it is possible to determine the overhead that is added to the per-frame MSDU payload when a frame is transmitted. There are at least the following sources of per-frame overhead:

- a) Any VLAN tags and security tags (see IEEE Std 802.1AE [B7]) that are added to the layer 2 payload as it passes through the various service interfaces in the Port’s protocol stack.
- b) The MAC framing (header and trailer octets, plus any padding octets that are required to meet minimum frame size limitations) that is added by the underlying MAC service.
- c) Any physical layer overhead, such as preamble characters and inter-frame gaps.

The precise per-frame overhead will therefore depend upon the protocol stack and the underlying MAC technology.

The actual bandwidth needed to support a given stream is therefore defined as follows [using *assumedPayloadSize* from Equation (1)]:

$$\text{actualBandwidth} = (\text{perFrameOverhead} + \text{assumedPayloadSize}) \times \text{maxFrameRate} \quad (3)$$

34.5 Mapping priorities to traffic classes for time-sensitive streams

In Bridges that support forwarding and queuing for time-sensitive streams, the default mappings of priorities to traffic classes meet the following constraints:

- a) Priority values that correspond to SR classes are mapped onto traffic classes that support the credit-based shaper algorithm as the transmission selection algorithm.
- b) Traffic classes that support the credit-based shaper algorithm have a higher priority than traffic classes that support the strict priority (or any other) transmission selection algorithm.
- c) At least one traffic class supports the credit-based shaper algorithm, and at least one traffic class supports the strict priority transmission selection algorithm.

NOTE 1—The constraint that there is at least one traffic class that supports the strict priority transmission selection ensures that there is at least one traffic class that can support traffic that is not subject to bandwidth reservation, such as “best effort” traffic.

The recommended default priority to traffic class mappings for a system that supports SR class A (using priority 3) and SR class B (using priority 2) are shown in Table 34-1. The recommended default priority to traffic class mappings for a system that supports only SR class B (using priority 2) are shown in Table 34-2.

The corresponding default configuration for the Transmission Selection Algorithm Table (see 8.6.8) is that the traffic classes that are shaded in the tables are configured to use the credit-based shaper algorithm, and the remaining traffic classes are configured to use the strict priority algorithm.

Table 34-1—Recommended priority to traffic class mappings for SR classes A and B

		Number of available traffic classes						
		2	3	4	5	6	7	8
Priority	0 (Default)	0	0	0	0	0	0	1
	1	0	0	0	0	0	0	0
	2	1	1	2	3	4	5	6
	3	1	2	3	4	5	6	7
	4	0	0	1	1	1	1	2
	5	0	0	1	1	1	2	3
	6	0	0	1	2	2	3	4
	7	0	0	1	2	3	4	5

Table 34-2—Recommended priority to traffic class mappings for SR class B only

		Number of available traffic classes						
		2	3	4	5	6	7	8
Priority	0 (Default)	0	0	0	0	0	1	1
	1	0	0	0	0	0	0	0
	2	1	2	3	4	5	6	7
	3	0	0	0	1	1	2	2
	4	0	1	1	2	2	3	3
	5	0	1	1	2	2	3	4
	6	0	1	2	3	3	4	5
	7	0	1	2	3	4	5	6

NOTE 2—The mapping shown in Table 34-1 for the case of only two available traffic classes maps two SR classes to a single traffic class. While this is a permissible configuration, in general it is desirable, and in some applications, can be a requirement, to assign SR classes to distinct traffic classes, as is done in this table for the cases where three or more traffic classes are available. The mappings shown deal only with one or two supported SR classes; a similar mapping strategy can be adopted if more than two SR classes are supported.

34.6 End station behavior

In order for an end station to successfully participate in the transmission and reception of time-sensitive streams, it is necessary for their behavior to be compatible with the operation of the forwarding and queueing mechanisms employed in bridges. The requirements for end stations that participate as “talkers”—i.e., sources of time-sensitive streams—are different from the requirements that apply to “listeners”—i.e., the destination station(s) for the streams.

34.6.1 Talker behavior

In order for Talker-originated data streams to make use of the credit-based shaper behavior in Bridges, it is a requirement for a Talker to use the priorities that the Bridges in the network recognize as being associated with SR classes exclusively for transmitting stream data. It is also necessary for the Talker, and the Bridges in the path to the Listener(s), to have a common view of the bandwidth required in order to transmit the Talker’s streams, and for that bandwidth to be reserved along the path to the Listener(s). This latter requirement can be met by means of stream reservation mechanisms, such as defined in SRP, or by other management means.

End stations that are Talkers shall exhibit transmission behavior for frames that are part of time-sensitive streams that is consistent with the operation of the credit-based shaper algorithm, both in terms of the way they transmit frames that are part of an individual data stream, and in terms of the way they transmit stream data frames from a Port. In effect, the queuing model for a Talker Port, and for a given priority, can be considered to look like Figure 34-1.

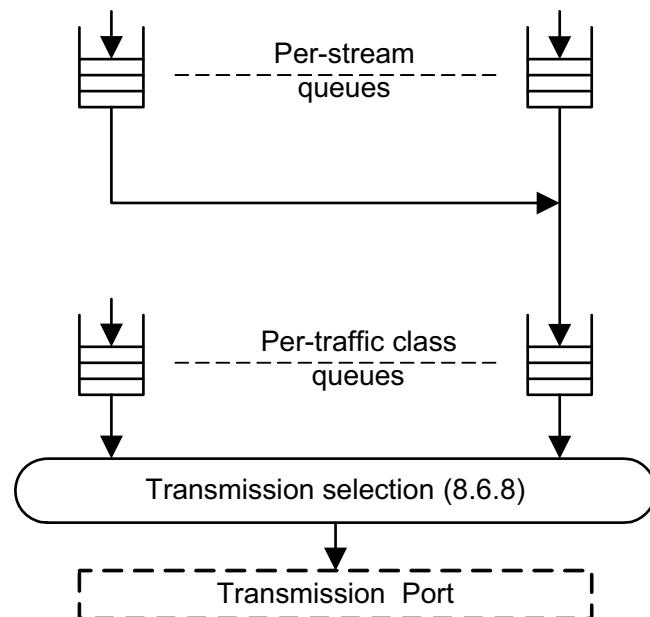


Figure 34-1—Queuing model for a Talker station

The Talker places frames into the queue associated with an individual stream based on the Tspec for that stream; i.e., during each class measurement interval, it can place up to *MaxIntervalFrames* data frames, each no longer than *MaxFrameSize* into that stream’s queue.

The queue associated with each individual stream uses the credit-based shaper algorithm, with the *idleSlope* set to the bandwidth requirement of the stream on that transmission Port, as the means of determining the rate at which data frames for that stream are placed in the outbound queue for the priority that the stream is

using. The outbound queue for that priority, in turn, makes use of the credit-based shaper algorithm, with the *idleSlope* set to the sum of the *idleSlope* values for all streams using that priority on that transmission Port, as the means of determining the rate at which data frames for that stream are selected for transmission.

Transmission selection in a Talker operates in the same way as in a Bridge; from this point of view, a Talker can be thought of as if it is a single-port Bridge.

For streams that make use of SR class A or SR class B, it is a requirement that the rate at which frames for any given stream are selected for placement in its per-stream queue does not exceed the bandwidth reserved for the stream, measured over the class measurement interval for the SR class (125 μ s for SR class A, 250 μ s for SR class B.) For some combinations of stream bandwidth requirement and transmission Port data rate, this can place a limit on the frame size that can be used when transmitting stream data.

NOTE—the sole implication of the observation interval is its effect on frame size, because the shaper behavior itself is independent of observation interval. The intent in limiting the observation interval is to limit frame size, as this is a major contribution to the latency experienced by a frame in transit through an AVB network (see discussion in Annex L).

34.6.2 Listener behavior

The primary requirement for a listener station is that it is capable of buffering the amount of data that could be transmitted for a stream during a time period equivalent to the accumulated maximum jitter that could be experienced by stream data frames in transmission between Talker and Listener. From the point of view of the specification of the forwarding and queuing requirements for time-sensitive streams, it is assumed that the listener will assess the buffering required for a stream as part of the stream bandwidth reservation mechanisms employed by the implementation.

35. Stream Registration Protocol (SRP)

The Stream Reservation Protocol (SRP) utilizes three signaling protocols, MMRP (10.9), MVRP (Clause 11) and MSRP (35.1) to establish stream reservations across a bridged network.

Within SRP the Multiple MAC Registration Protocol (MMRP) is optionally used to control the propagation of Talker registrations throughout the bridged network (35.2.4.3.1).

The Multiple VLAN Registration Protocol (MVRP) is used by end stations and Bridges to declare membership in a VLAN where a Stream is being sourced. This allows the Data Frame Priority (35.2.2.8.5(a)) to be propagated along the path from Talker to Listener(s) in tagged frames. MSRP will not allow Streams to be established across Bridge Ports that are members of the untagged set (8.8.10) for the related VLAN ID.

The Multiple Stream Registration Protocol (MSRP) is a signaling protocol that provides end stations with the ability to reserve network resources that will guarantee the transmission and reception of data streams across a network with the requested quality of service. These end stations are referred to as Talkers (devices that produce data streams) and Listeners (devices that consume data streams).

Talkers declare attributes that define the stream characteristics so Bridges have the necessary information available when they need to allocate resources for a Stream. Listeners declare attributes that request reception of those same streams. Bridges along the path from Talker to Listener process, possibly adjust, and then forward these MSRP attribute declarations. Bridges associate Talker and Listener attributes via the StreamID present in each of those attributes, which result in changes to the extended filtering services and allocation of internal resources when streams are “brought up.”

In order to establish the SRP domain boundaries, Bridges exchange SR class characteristics with each other and with end stations via MSRP. Neighboring devices that have identical SR class characteristics are considered to be in the same SRP domain and streams may be established between those devices.

MSRP provides a limited error reporting capability that is utilized when a Listener’s request to receive a stream cannot be honored because of some resource constraint within the network.

MSRP also supports the concept of data stream importance. For example, an emergency announcement would be flagged with a more important “rank” than a stream providing background music. This ranking ability allows the bridges to replace less important streams with more important streams without requiring intervention from the end stations.

There is a considerable body of experience in supplying data streams with guarantees for quality of service parameters such as latency, latency variation, or bandwidth. In particular, routers and hosts use the Internet Protocol and the Resource Reservation Protocol (RSVP, IETF RFC 2205 and RFC 2750) to achieve such guarantees. Supplying guarantees to a data stream requires the following two components:

- a) A definition of the resources to be allocated and configured, by end stations and network nodes, for the support of a data stream; and
- b) A protocol for end stations to signal to the network nodes their data streams’ requirements, for network nodes to distribute those requirements among each other, and for the network nodes to signal the success or failure of the attempt to reserve resources to support the guarantees.

RSVP supplies the signaling protocol for routers to support data streams in routed networks. This and the following clauses define a protocol to support data streams in bridged networks.

35.1 Multiple Stream Registration Protocol (MSRP)

MSRP supports the reservation of resources for streams, each destined for one or more Listeners, and each from a single source, across a bridged network. Transmitted data that conforms to a successful stream reservation will not be discarded by any Bridge due to congestion on a LAN. In order to propagate requests for reservations, MSRP defines an *MRP application* that provides the Stream resource registration service defined in 35.2.3. MSRP makes use of the MRP Attribute Declaration (MAD) function, which provides the common state machine descriptions defined for use in MRP-based applications. The MRP architecture, and MAD are defined in Clause 10. MSRP defines a new MRP Attribute Propagation (MAP) function, to provide an attribute propagation mechanism.

MSRP propagates registrations for stream reservations in a manner similar to the operation of MMRP (10.9) and MVRP (11.2), which are used for registering Group membership and individual MAC address information, and VLAN membership, respectively. Unlike MMRP and MVRP, however, the registered attributes can be combined, discarded, or otherwise altered, as they are propagated by the participating Bridges.

In order to make and keep quality of service guarantees all devices in a bridged network must participate in the signaling and queuing operations required of Bridges. For example, this would include IEEE 802.11 wireless media access points and stations. Thus, MSRP provides a means for Bridges or end stations running MSRP to cooperate both with higher network layers, such as routers or hosts running RSVP, and with lower network layers, such as wireless media.

MSRP is also responsible for establishing the SRP domain boundary for a particular SR class. All systems that support a particular SR class are in the same SRP domain if they use the same priority. An SRP domain boundary exists for an SR class when neighboring devices use different priorities for the SR class.

Figure 35-1 illustrates the architecture of MSRP in the case of a two-Port Bridge and an end station.

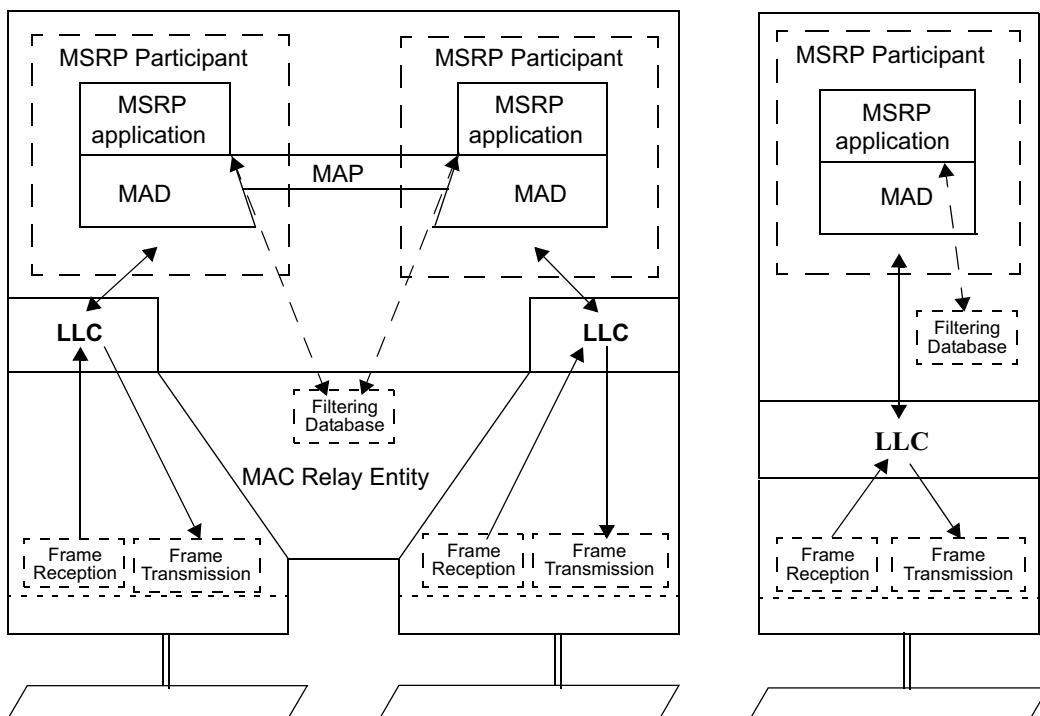


Figure 35-1—Operation of MSRP

35.1.1 MSRP and Shared Media

Classic shared media, such as IEEE 802.3 half-duplex Carrier Sense Multiple Access with Collision Detect (CSMA/CD), cannot provide latency or bandwidth guarantees, because their operation depends on random timers. Such media are, therefore, not supported by MSRP.

There are other shared media where one node on the medium exercises control over access to the medium by the other nodes. For example, an IEEE 802.11 wireless medium has a single Access Point (AP) that controls access by the AP and the stations attached to the wireless medium so that some guarantee of latency and bandwidth can be made, subject to frame loss caused by data corruption errors. Similarly, an IEEE 802.3 Ethernet Passive Optical Network has a single Optical Line Terminal (OLT) that controls access to the optical medium by itself and some number of Optical Network Units (ONUs).

Different kinds of shared media use different techniques to allocate opportunities to transmit, and these techniques can have various dependencies on frame sizes, station-to-station vs. station-to-head data paths, or other factors. Rather than introducing the complexities of every such medium into MSRP, this standard takes advantage of the presence of a controlling entity to map medium-specific characteristics to the capabilities of MSRP.

MSRP defines and requires the existence of a Designated MSRP Node (DMN) on any shared medium. This DMN provides the MSRP services for the shared medium and determines each station's ability to receive the MSRPDU transmitted by other stations on the medium. A Non-DMN Port shall be configured to only process the MSRPDU transmitted by DMN Ports, and ignore the MSRPDU transmitted by other Non-DMN Ports. Furthermore, the DMN has the absolute control of the resource allocation on the shared medium. Given these two facts, a DMN can effectively control which reservations are and are not successful on the medium it controls.

Annex C provides examples on various shared media.

35.1.2 Behavior of end stations

35.1.2.1 Talkers

To announce the Streams that can be supplied and their characteristics, Talkers use the MAD_Join.request primitive (10.2) to make the Talker Declarations (35.2.1.3). To indicate the Streams that are no longer supplied, Talkers use the MAD_Leave.request primitive (10.2) to withdraw their Talker Declarations.

Talker Declarations are propagated by MSRP such that the Listeners and Bridges are aware of the presence of Talkers and the Streams that are offered. Talker Declarations are also used to gather QoS information along their paths. Based on the gathered QoS information, Talker Declarations are classified as follows:

- a) **Talker Advertise:** An advertisement for a Stream that has not encountered any bandwidth or other network constraints along the network path from the Talker. Listeners that request attachment to this Stream are likely to create a reservation with the described QoS. A Talker Advertise will continue to be declared as long as the resources continue to be available.
- b) **Talker Failed:** An advertisement for a Stream that is not available to the Listener because of bandwidth constraints or other limitations somewhere along the path from the Talker.

Talkers respond to the registration and de-registration events of Listener Declarations (35.2.1.3), signalled by MAD as follows:

On receipt of a MAD_Join.indication for a Listener Declaration, the Talker first merges (35.2.4.4.3) the Listener Declarations that it has registered for the same Stream. Then the Talker examines the StreamID (35.2.2.8.2) and Declaration Type (35.2.1.3) of the merged Listener Declaration. If the merged Listener

Declaration is associated with a Stream that the Talker can supply, and the DeclarationType is either Ready or Ready Failed (i.e. one or more Listeners can receive the Stream), the Talker can start the transmission for this Stream immediately. If the merged Listener Declaration is an Asking Failed, the Talker shall stop the transmission for the Stream, if it is transmitting.

On receipt of a MAD_Leave.indication for a Listener Declaration, if the StreamID of the Declaration matches a Stream that the Talker is transmitting, then the Talker shall stop the transmission for this Stream, if it is transmitting.

35.1.2.2 Listeners

To indicate what Streams they want to receive, Listeners use the MAD_Join.request primitive (10.2) to make the Listener Declarations (35.2.1.3). To indicate the Streams that are no longer wanted, Listeners use the MAD_Leave.request primitive (10.2) to withdraw their Listener Declarations.

The Listener Declaration also conveys the results of the bandwidth and resource allocation along its path back to the Talker. Based on those results, Listener declarations are classified as follows:

- a) **Listener Ready:** One or more Listeners are requesting attachment to the Stream. There is sufficient bandwidth and resources available along the path(s) back to the Talker for all Listeners to receive the Stream.
- b) **Listener Ready Failed:** Two or more Listeners are requesting attachment to the Stream. At least one of those Listeners has sufficient bandwidth and resources along the path to receive the Stream, but one or more other Listeners are unable to receive the stream because of network bandwidth or resource allocation problems.
- c) **Listener Asking Failed:** One or more Listeners are requesting attachment to the Stream. None of those Listeners are able to receive the Stream because of network bandwidth or resource allocation problems.

NOTE—The reader will notice that the Talker response to Ready and Ready Failed declarations is the same: the Talker can begin transmitting the Stream. Talkers might choose to pass the Ready Failed response to a higher layer protocol that could notify the user that the Stream is flowing, but not all Listeners are receiving it. It would be the responsibility of that higher layer to respond to this information as appropriate.

When there is a Talker Declaration registered on an interested Listener end station, the Listener shall create a Listener Declaration as follows:

If the Listener receives a Talker Advertise declaration, and the Listener is ready to receive the Stream, the Listener shall declare the following in the order specified:

- 1) An MVRF VLAN membership request for the vlan_identifier contained in the Talker Advertise DataFrameParameters [35.2.2.8.3(b)] so the neighboring bridge will add the associated Bridge Port to the member set for the VLAN;
- 2) An MSRP Listener Ready declaration for the Stream.

If the Listener receives a Talker Failed declaration, and the Listener is ready to receive the Stream, the Listener shall issue a Listener Asking Failed declaration for the Stream.

There is no requirement for the order in which the Talker and Listener declarations are communicated. The Listener Declaration can be made before the Listener receives an associated Talker Declaration, in which case the Listener shall issue a Listener Asking Failed declaration.

35.1.3 Behavior of Bridges

MSRP-aware Bridges register and de-register Talker and Listener declarations on the Bridge Ports according to the procedures defined in MRP (Clause 10), and automatically generate de-registration of stale registrations. Any changes in the state of registration are processed by the MSRP Attribute Propagation (35.2.4) function, and disseminated in the network by making or withdrawing Talker and Listener declarations as defined in the Talker attribute propagation (35.2.4.3) and Listener attribute propagation (35.2.4.4).

In general, Talker declarations are propagated to all other Bridge Ports. There is a *talkerPruning* option (35.2.1.4(b)) that limits the scope of Talker declaration propagation. Listener declarations are only propagated to the Bridge Port with the associated Talker declaration (i.e. matching StreamID). If there is no associated Talker declaration registered on any Bridge Port then Listener declaration will not be propagated.

35.1.3.1 Blocked Declarations

For the purposes of MSRP Attribute Propagation (35.2.4), a Declaration is said to be “blocked” on a Bridge Port if the state of the Spanning Tree Instance identified by the *vlan_identifier* in the *DataFrameParameters* (35.2.2.8.3) of the Declaration, on that Bridge Port, has any value other than Forwarding. In a station’s Participant, no Declaration is ever blocked.

35.2 Definition of the MSRP application

MSRP maintains two categories of variables. The first category is used internally by the application state machines. These are defined in detail in the subclauses that follow.

MSRP also defines another category of variables identified as MRP elements that are communicated in MSRPDUs between stations on a network. These protocol elements include the MRP frame addressing and other fields defined in the MRP Protocol Data Units. The MSRP FirstValue fields, which are used to exchange the MSRP attributes, are also defined here.

35.2.1 Definition of internal state variables

The following variables and parameters are utilized by various state machines within MSRP:

- a) Port Media Type (35.2.1.1);
- b) Direction (35.2.1.2);
- c) Declaration Type (35.2.1.3);
- d) SRP parameters (35.2.1.4);

35.2.1.1 Port Media Type

MSRPDU processing on a port is handled differently depending on the type of medium the port is attached to. For example, the DMN on a shared medium port that receives MSRPDUs from one station shall update and retransmit those attributes so that all stations on that medium are updated appropriately. The possible values are as follows:

- a) **Access Control Port:** Transmitter controls access to the medium on which it is sending, so either it is the DMN for a shared medium, or it is a port on a full-duplex point-to-point medium;
- b) **Non-DMN shared medium Port:** Transmitter is attached to a shared medium, but does not control access to the medium.

35.2.1.2 Direction

The Direction field is derived from the MSRP AttributeType definitions (35.2.2.4). The Direction indicates whether this is a Talker or a Listener MSRP Declaration, and takes one of the following two values:

- a) **Talker:** MSRP AttributeType definitions of type Talker Advertise Vector Attribute Type (35.2.2.4(a)) or Talker Failed Vector Attribute Type (35.2.2.4(b)). Set Direction to zero for Talker attributes.
- b) **Listener:** MSRP AttributeType definitions of type Listener Vector Attribute Type (35.2.2.4(c)). Set Direction to one for Listener attributes.

35.2.1.3 Declaration Type

The Declaration Type field is derived from the MSRP AttributeType definitions (35.2.2.4) and the MSRP FourPackedEvents (35.2.2.7.2). The Declaration Type indicates the specific type of the Talker or Listener MSRP Declaration.

For a Talker, the value of the Declaration Type component is either:

- a) **Advertise:** MSRP AttributeType definitions of Talker Advertise Vector Attribute Type (35.2.2.4(a)).
- b) **Failed:** MSRP AttributeType definitions of Talker Failed Vector Attribute Type (35.2.2.4(b)).

For a Listener, the value of the Declaration Type component is one of the following:

- c) **Asking Failed:** MSRP AttributeType definitions of Listener Vector Attribute Type (35.2.2.4(c)) with MSRP FourPackedType equal to Asking Failed (35.2.2.7.2(b)).
- d) **Ready:** MSRP AttributeType definitions of Listener Vector Attribute Type with MSRP FourPackedType equal to Ready (35.2.2.7.2(c)).
- e) **Ready Failed:** MSRP AttributeType definitions of Listener Vector Attribute Type with MSRP FourPackedType equal to Ready Failed (35.2.2.7.2(d)).

35.2.1.4 SRP parameters

The following parameters are used by SRP:

- a) **portTcMaxLatency:** The maximum per-port per-traffic class latency, expressed in nanoseconds, a frame may experience through the underlying MAC service. There may be different latency numbers for different traffic classes on the same port.
- b) **talkerPruning:** Enabling this parameter on the Bridge will limit the Talker declarations to ports that have the Streams destination_address (35.2.2.8.3(a)) in the MMRP MAC Address Registration Entries.
- c) **streamAge:** A per-port per-stream 32-bit unsigned value used to represent the time, in seconds, since a stream's destination_address was first added to the Dynamic Reservations Entries (8.8.7) for the associated port. This value is used when determining which streams have been configured the longest. Streams with a numerically larger *streamAge* are considered to be configured earlier than other streams, and therefore carry a higher implicit importance.

NOTE 1—A 32-bit unsigned value allows for expressing a *streamAge* of up to 136 years.

- d) **msrpEnabledStatus:** MSRP shall have the ability to be enabled (true) or disabled (false) on a device. When MSRP is enabled on a device it shall cause a reset of all MSRP state machines on all ports. This affects the Applicant and Registrar state machines. The state of this parameter shall be persistent over power-up restart/reboot.

- e) **msrpPortEnabledStatus:** MSRP shall have the ability to be enabled (true) or disabled (false) on the ports of a device. When MSRP is enabled or disabled on a port of a device it shall cause MAP to be rerun on all MSRP enabled ports so existing attributes can be propagated to the port just enabled. This affects the Applicant and Registrar state machines. The state of this parameter shall be persistent over power-up restart/reboot.
- f) **msrpMaxFanInPorts:** The total number of ports on a Bridge that are allowed to establish reservations for inbound Streams. This number may be less than the total number of ports with msrpPortEnabledStatus set TRUE, which will result in lower maximum latency because of limits on the amount of possible interfering traffic. A value of zero (0) indicates no fan-in limit is being specified and calculations involving fan-in will only be limited by the number of MSRP enabled ports. Example calculations of delay associated with fan-in can be found in the paper “Calculating the Delay Added by Qav Stream Queue” [B4].
- g) **msrpLatencyMaxFrameSize:** Calculation of the maximum latency through a bridge is in-part related to the maximum size of an interfering frame. The maximum size is defined to be 2000 octets by default. This parameter allows a smaller or larger value to be used in the latency calculations for the particular Bridge implementation. msrpLatencyMaxFrameSize does not imply any type of policing of frame size, it is only used in the latency calculations.
- h) **SRPdomainBoundaryPort:** A per-port, per-SR class, Boolean parameter that contains the value TRUE if the port is an SRP Domain Boundary Port, otherwise it contains the value FALSE. The parameter for a given SR class and Port shall be set to TRUE if any of the following conditions are met:
 - 1) The port is declaring an MSRP Domain attribute for that SR class, and the port has no MSRP Domain attribute registrations for that SR class, or;
 - 2) The port is declaring an MSRP Domain attribute for that SR class, and the port has at least one MSRP Domain attribute registration for that SR class with a different priority, or;
 - 3) One or more ports which support that SR class are declaring MSRP Domain attributes for that SR class, and this port does not support that SR class.In all other cases the parameter shall be set to FALSE.
- i) **SR_PVID:** The Stream Reservation Port VLAN Identifier (SR_PVID) is a per-port parameter that contains the default VLAN ID for Stream related traffic. It shall contain a valid VID value (Table 9-2) and may be configured by management. If the value has not been explicitly configured, the SR_PVID shall assume the default SR_PVID defined in Table 9-2. This value is passed to the Talker via the SRclassVID (35.2.2.9.4) contained in the MSRP Domain attribute.

35.2.2 Definition of MRP elements

35.2.2.1 MSRP application address

The group MAC address used as the destination address for MRPDUs destined for MSRP Participants shall be the group MAC address for “Individual LAN Scope group address, Nearest Bridge group address” as specified in Table 8-1, Table 8-2 and Table 8-3 (C-VLAN, S-VLAN and TPMR component Reserved addresses, respectively).

NOTE—Using this address will guarantee that MRPDUs are never forwarded by an 802.1 Bridge, although MSRP aware Bridges do propagate the MSRP attributes.

35.2.2.2 MSRP application EtherType

The EtherType used for MRPDUs destined for MSRP Participants shall be the MSRP EtherType identified in Table 10-2.

35.2.2.3 MSRP ProtocolVersion

The ProtocolVersion for the version of MSRP defined in this standard takes the hexadecimal value 0x00.

35.2.2.4 MSRP AttributeType definitions

MSRP defines four AttributeTypes (10.8.2.2) that are carried in MRP exchanges. The numeric values for the AttributeType are shown in Table 35-1 and their use is defined by the following list:

- a) **Talker Advertise Vector Attribute Type:** Attributes identified by the Talker Advertise Vector Attribute Type are instances of VectorAttributes (10.8.1), used to identify a sequence of values of Talker advertisements for related Streams that have not been constrained by insufficient bandwidth or resources.
- b) **Talker Failed Vector Attribute Type:** Attributes identified by the Talker Failed Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values of Talker advertisements for related Streams that have been constrained by insufficient bandwidth or resources.
- c) **Listener Vector Attribute Type:** Attributes identified by the Listener Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values of Listener requests for related Streams regardless of bandwidth constraints. Listener Vector Attribute Types are subdivided into individual Declaration Types via the MSRP FourPackedEvents (35.2.2.7.2).
- d) **Domain Vector Attribute Type:** Attributes identified by the Domain Vector Attribute Type are instances of VectorAttributes, used to identify a sequence of values that describe the characteristics of an SR class.

Table 35-1—AttributeType Values

AttributeType	Value
Talker Advertise Vector	1
Talker Failed Vector	2
Listener Vector	3
Domain Vector	4

35.2.2.5 MSRP AttributeLength definitions

The AttributeLength field (10.12.1.8) in instances of the Talker Advertise Vector Attribute Type shall be encoded in MRPDUs (10.8) as an unsigned binary number, equal to the value shown in Table 35-2 (AttributeLength Values).

Table 35-2—AttributeLength Values

AttributeType	Value
Talker Advertise Vector	25 (0x19)
Talker Failed Vector	34 (0x22)
Listener Vector	8
Domain Vector	4

The AttributeLength field in instances of the Talker Failed Vector Attribute Type shall be encoded in MRPDUs as an unsigned binary number, equal to the value shown in Table 35-2.

The AttributeLength field in instances of the Listener Vector Attribute Type shall be encoded in MRPDUs as an unsigned binary number, equal to the value shown in Table 35-2.

The AttributeLength field in instances of the Domain Vector Attribute Type shall be encoded in MRPDUs as an unsigned binary number, equal to the value shown in Table 35-2.

35.2.2.6 MSRP AttributeListLength definitions

The AttributeListLength field (10.12.1.9) shall be encoded in MRPDUs (10.8) as an unsigned binary number, equal to the number of octets contained within the AttributeList. This field can be used when calculating the number of octets to skip to proceed to the next Message (or Message EndMark) in the MRPDU.

35.2.2.7 MSRP Vector definitions

35.2.2.7.1 MSRP ThreePackedEvents

The ThreePackedEvent vectors are encoded as defined in 10.8.2.10.1.

35.2.2.7.2 MSRP FourPackedEvents

MSRP FourPackedEvents are only used by the Listener Vector Attribute Type (35.2.2.4(c)). Within the FourPackedEvent, there are four possible values for the FourPackedType (10.8.2.10.2) as explained in the following list. The numeric values for the MSRP FourPackedEvents are shown in Table 35-3.

- a) **Ignore:** The StreamID referenced by FirstValue+n is not defined in this MSRPDU. When using this FourPackedType, the AttributeEvent (10.8.2.10.1) value, encoded in the ThreePackedEvent, shall be set to zero on transmit and ignored on receive
- b) **Asking Failed:** The StreamID referenced by FirstValue+n has a declaration type of Listener Asking Failed.
- c) **Ready:** The StreamID referenced by FirstValue+n has a declaration type of Listener Ready.
- d) **Ready Failed:** The StreamID referenced by FirstValue+n has a declaration of type Listener Ready Failed.

Table 35-3—FourPackedEvent Values

FourPackedType	Value
Ignore	0
Asking Failed	1
Ready	2
Ready Failed	3

NOTE—In terms of efficient use of octets within an MSRP packet, placing nineteen (19) Ignores between two Listener declarations uses less octets than using two VectorAttributes to declare the Listener attributes separately. A single Listener VectorAttribute takes 12 octets, two attributes would take 24 octets. Declaring 21 attributes (two valid attributes with 19 Ignores inbetween) takes 12 octets, plus 6 additional ThreePackedEvents, plus 5 additional FourPackedEvents for a total of 23 octets.

35.2.2.8 MSRP FirstValue definitions (Stream reservations)

There are four Attribute Declarations defined for MSRP (35.2.2.4), three of which are related to stream reservations: Talker Advertise, Talker Failed, and Listener. The fourth attribute type, Domain, is used to discover the SRP domain, and is described in clause 35.2.2.9.

The Talker Advertise attribute contains all the characteristics that a Bridge needs in order to understand the resource requirements and importance of the referenced stream.

The Talker Failed attribute contains all the fields carried in the Talker Advertise Attribute, plus additional information regarding resource or bandwidth availability failures.

Listener attributes carry three subtypes: Ready, Ready Failed, and Asking Failed. These Listener subtypes are encoded in the FourPackedEvent (35.2.2.7.2).

FirstValue shall be incremented one or more times when NumberOfValues is greater than one. Incrementing FirstValue within MSRP is defined as follows:

- a) Add 1 to Unique ID (35.2.2.8.2(b)), and
- b) Add 1 to Stream destination_address (35.2.2.8.3(a))

The example shown in Table 35-4 illustrates the use of FirstValue and NumberOfValues within MSRP. This example shows four Streams (a, b, c and d) to be declared. In order to use the efficient packing techniques of MRP it would be preferable to assign these Streams sequential StreamIDs and destination_addresses as shown. Notice that StreamID yy-yy-yy-yy-yy-yy:00-04 is missing from this table (between Stream “c” and “d”). MSRP allows declaration of all four StreamIDs in a single VectorAttribute by setting NumberOfValues=5, with the StreamID = yy-yy-yy-yy-yy-yy:00-01 and a destination_address of xx-xx-xx-xx-xx-25. Setting the fourth FourPackedEvent to Ignore (35.2.2.7.2(a)) notifies MSRP that the StreamID between “c” and “d” is not in use.

Table 35-4—MSRP FirstValue NumberOfValues example

Stream	StreamID	destination_address
a	yy-yy-yy-yy-yy-yy:00-01	xx-xx-xx-xx-xx-25
b	yy-yy-yy-yy-yy-yy:00-02	xx-xx-xx-xx-xx-26
c	yy-yy-yy-yy-yy-yy:00-03	xx-xx-xx-xx-xx-27
d	yy-yy-yy-yy-yy-yy:00-05	xx-xx-xx-xx-xx-29

The FirstValue field within MSRP is comprised of several components: StreamID (35.2.2.8.2), DataFrameParameters (35.2.2.8.3), TSpec (35.2.2.8.4), PriorityAndRank (35.2.2.8.5), Accumulated Latency (35.2.2.8.6), and FailureInformation (35.2.2.8.7). MSRP does not support changes in any of the FirstValue fields for an existing StreamID. If a Talker wishes to tear an old Stream down and bring a new Stream up,

with a different FirstValue, utilizing the same StreamID, there must be at least two LeaveAllTime (Table 10-7) time periods between when the Talker removes the existing Stream registration and declares the new Stream. This guarantees that MRP has enough time to remove the current attribute from all devices in the network. If the new declaration were to occur too quickly the associated Streaming data could be corrupted because the filtering database may allow the new Stream data to start flowing while the old Stream bandwidth constraints are still configured. When the Bridge detects this occurring it will fail the Talker Advertise with the appropriate FailureInformation (35.2.2.8.7). Talkers may tear a Stream down and bring the same Stream back up immediately, as long as the FirstValue has not changed.

35.2.2.8.1 Structure definition

The FirstValue for Talker Advertise, Talker Failed and Listener attributes in MSRPDUs exchanged according to the protocol specified in this subclause shall have the following structure:

- a) The first eight octets contain the *StreamID* (35.2.2.8.2).

This is the end of the Listener attribute. If this is a Talker Advertise or Talker Failed attribute continue as follows:

- b) Following the StreamID are eight octets containing the *DataFrameParameters* (35.2.2.8.3).
- c) Following the DataFrameParameters are four octets containing the *TSpec* (35.2.2.8.4).
- d) Following the TSpec is one octet containing the *PriorityAndRank* (35.2.2.8.5).
- e) Following the PriorityAndRank are four octets containing the *AccumulatedLatency* (35.2.2.8.6).

This is the end of the Talker Advertise attribute. If this is a Talker Failed attribute continue as follows:

- f) Following the AccumulatedLatency are nine octets containing the *FailureInformation* (35.2.2.8.7).

The following partial BNF production gives the formal description of the MSRPDU FirstValue structure for the Talker Advertise attribute:

```
FirstValue ::= StreamID, DataFrameParameters, TSpec, PriorityAndRank, AccumulatedLatency
```

The following partial BNF production gives the formal description of the MSRPDU FirstValue structure for the Talker Failed attribute:

```
FirstValue ::= StreamID, DataFrameParameters, TSpec, PriorityAndRank, AccumulatedLatency,  
FailureInformation
```

The following partial BNF production gives the formal description of the MSRPDU FirstValue structure for the Listener attribute:

```
FirstValue ::= StreamID
```

Figure 35-2 illustrates the structure of the MSRPDU FirstValue components. Each MSRP Attribute shall only use those structures as defined by the partial BNF productions described previously. The octet numbers shown represent the octet location within the FirstValue field.

Octet #				
1		7	8	
MAC Address		Unique ID		StreamID structure
Octet #				
9		15	16	
destination_address		vlan_identifier		DataFrameParameters structure
Octet #				
17		19	20	
MaxFrameSize		MaxIntervalFrames		TSpec structure
Octet #				
21				
Data Frame Priority (3 bits)	Rank (1 bit)	Reserved (4 bits)		PriorityAndRank structure
Octet #				
22		25		
Accumulated Latency				AccumulatedLatency structure
Octet #				
26		34		
Bridge ID		Failure Code		FailureInformation structure

Figure 35-2—Format of the components of the reservation FirstValue fields

35.2.2.8.2 StreamID

The 64-bit StreamID is used to match Talker registrations with their corresponding Listener registrations (35.2.4). The StreamID comprises the following two subcomponents:

- a) An EUI-48 MAC Address associated with the System sourcing the stream to the bridged network. The entire range of EUI-48 addresses are acceptable.
- b) A 16-bit unsigned integer value, Unique ID, used to distinguish among multiple streams sourced by the same System.

StreamIDs are unique across the entire bridged network and are generated by the system offering the stream, or possibly a device controlling that system. A system reserving resources for more than one stream in the same bridged network shall use a StreamID that is unique among all StreamIDs in that bridged network. The combination of these two subcomponents ensure that such an assignment is possible.

NOTE 1—The Spanning Tree Protocol ensures that there can be at most one path from a Talker to its Listener(s). Multiple Declarations for the same StreamID can therefore occur briefly, during changes in the active topology of the bridged network.

NOTE 2—The MAC address component of the StreamID can, but does not necessarily, have the same value as the source_address parameter of any frame in the actual data stream.

35.2.2.8.3 DataFrameParameters

The DataFrameParameters component of the MSRP Attribute specifies the EISS parameters that are common to all frames belonging to the data stream for which this MAD is reserving resources. This information is used by Bridges to create Dynamic Reservation Entries (8.8.7). The parameters are as follows:

- a) The destination_address; and
- b) The vlan_identifier.

The destination_address specifies the destination MAC address of the streaming data packets. Only one Talker is allowed per destination_address. MSRP does not describe the actual streaming data, only the bandwidth associated with that stream.

The use of destination_address for both a Stream and “best effort” traffic (34.5) is outside the scope of SRP. SRP only supports destination_addresses that are multicast or locally administered addresses.

NOTE—MSRP enforces reserved bandwidth guarantees by filtering Stream destination addresses (35.2.4.4.2) for Streams that do not have a reservation. This blocks all traffic to that destination address. If that destination address was also being used for “best effort” traffic that device would no longer be reachable.

Systems that are not VLAN aware shall use the value SRclassVID (35.2.2.9.4) for the vlan_identifier in the DataFrameParameters. VLAN aware systems may use any valid VID (1 through 4094).

35.2.2.8.4 TSpec

The 32-bit TSpec component is the Traffic Specification associated with a Stream. It consists of the following two elements (which are encoded as described in 10.8.1.1):

- a) **MaxFrameSize:** The 16-bit unsigned MaxFrameSize component is used to allocate resources and adjust queue selection parameters in order to supply the quality of service requested by an MSRP Talker Declaration. It represents the maximum frame size that the Talker will produce, excluding any overhead for media specific framing (e.g., preamble, IEEE 802.3 header, Priority/VID tag, CRC, interframe gap). As the Talker or Bridge determines the amount of bandwidth to reserve on the egress port it will calculate the media specific framing overhead on that port and add it to the number specified in the MaxFrameSize field.
- b) **MaxIntervalFrames:** The 16-bit unsigned MaxIntervalFrames component is used to allocate resources and adjust queue selection parameters in order to supply the quality of service requested by an MSRP Talker Declaration. It represents the maximum number of frames that the Talker may transmit in one “class measurement interval” (34.4).

NOTE—Consider the example of a Class A 48kHz stereo audio stream encapsulated in an ethernet frame [B11]. The audio data within the frame would contain two sets of six 32-bit samples, plus a 32-octet header, for a total of 80 octets per frame sent once every class measurement interval (34.4). Therefore, MaxFrameSize=80, and MaxIntervalFrames=1. An IEEE 802.3 port on a Bridge would also add 42 octets of media specific framing overhead (8-octet preamble, 14-octet IEEE 802.3 header, 4-octet IEEE 802.1Q priority/VID Tag, 4-octet CRC, 12-octet IFG). When the Bridge calculates the amount of bandwidth to reserve it would combine 42 octets of media specific framing overhead with the MaxFrameSize of 80 octets, to arrive at a total frame size of 122 octets per class measurement interval. This represents a total bandwidth of approximately 7.7Mbit/s (122 octets * 8 bits/octet * 8000 frames/s).

Table 35-5 contains some examples of various forms of audio and video Streams with their associated TSpec components.

Table 35-5—TSpec Components Examples

Source	Raw Bit Rate	Media Specific Framing Overhead	TSpec MaxFrameSize	TSpec MaxIntervalFrames
48kHz stereo audio stream (32-bit samples) Class A [B11]	~3 Mbit/sec	~4.7 Mbit/sec	80	1 (8,000 frames/sec)
96kHz stereo audio stream (32-bit samples) Class A [B11]	~6 Mbit/sec	~4.7 Mbit/sec	128	1 (8,000 frames/sec)
MPEG2-TS video Class B [B11]	~24 MBit/sec	~2.5 Mbit/sec	786	1 (4,000 frames/sec) ¹
SD SDI (Level C) uncompressed Class A [B46]	270 Mbit/sec	~15 Mbit/sec	1442	3 (24,000 frames/sec)
SD SDI (Level D) uncompressed Class A [B46]	360 Mbit/sec	~20 Mbit/sec	1442	4 (32,000 frames/sec)
HD SDI 1080i uncompressed Class A [B47]	1.485 Gbit/sec	~80 Mbit/sec	1486	16 (128,000 frames/sec)
HD SDI 1080p uncompressed Class A [B48]	2.97 Gbit/sec	~160 Mbit/sec	1486	32 (256,000 frames/sec)

¹The MPEG-2 TS entry in this table (third row) is shown as Class B traffic which runs at a default frame rate of 250 µs/frame, or 4000 frames/sec. Class A traffic runs at a default rate of 8000 frames/sec.

35.2.2.8.5 PriorityAndRank

- a) **Data Frame Priority:** The 3-bit Data Frame Priority component specifies the priority value used to generate the Priority Code Point (PCP) the referenced data streams will be tagged with. It indicates the priority EISS or ISS parameter that will be used in all frames belonging to the data stream for which this MAD is reserving resources. This parameter determines which queue the frame is placed into on an output Bridge Port. In accordance with 10.8.1.1 these three bits are in the most significant positions (8, 7 and 6). The priority specified here is associated with the SR Classes as described in 34.5.
- b) **Rank:** The single-bit Rank component is used by systems to decide which streams can and cannot be served, when the MSRP registrations exceed the capacity of a Port to carry the corresponding data streams. If a Bridge becomes oversubscribed (e.g., network reconfiguration, IEEE 802.11 bandwidth reduction) the Rank will also be used to help determine which Stream or Streams can be dropped. A lower numeric value is more important than a higher numeric value. In accordance with 10.8.1.1 this bit is in position 5.

For streams that carry emergency data such as North America 911 emergency services telephone calls, or fire safety announcements, the Rank shall be 0. Nonemergency traffic shall set this bit to a 1.

NOTE—It is expected that higher layer applications and protocols can use the Rank to indicate the relative importance of streams based on user preferences expressed by means beyond the scope of this standard. The values and defaults provided by this Standard are sufficient to order streams on a first-come-first-served basis, with special priority provided for emergency services.

- c) **Reserved:** This 4-bit field shall be zero filled on transmit and ignored on receive. In accordance with 10.8.1.1 these four bits are in the least significant positions (4, 3, 2 and 1).

35.2.2.8.6 Accumulated Latency

The 32-bit unsigned Accumulated Latency component is used to determine the worst-case latency that a Stream can encounter in its path from the Talker to a given Listener. The latency reported here is not intended to increase during the life of the reservation. If some event occurs that would increase the latency beyond the original guarantee, MSRP will change the Talker Advertise to a Talker Failed and report Failure Code=7 (Table 35-6).

NOTE 1—An example of how latency could increase is if the speed of the underlying media were to decrease, such as one might see on a wireless link.

The initial value sent by the Talker is set to *portTcMaxLatency*: plus any amounts specified in the REGISTER_STREAM.request, and its value is increased by each Bridge as the Talker Declaration propagates through the network.

The *portTcMaxLatency*: per hop is equal to the sum of:

- a) (equal or higher priority traffic) the time required to empty the queue in which frames of that priority are placed, if that queue and all higher priority queues are full;
- b) (lower priority traffic) the time required to transfer one lower priority frame of maximum size that could have just started transmitting as the current priority frame was queued up;
- c) (internal processing) the worst-case time required by the Bridge to transfer a received frame from the input port to the output queue; and
- d) (wire propagation time) the time required for the first bit of the frame to propagate from the output port to the receiving device;
- e) (media access delay) the time required to wait for the media to become available for transmission.

For item a) the total number of ports with *msrpPortEnabledStatus* set TRUE, and the *msrpMaxFanInPorts* will effect these calculations⁴⁵.

For item a) and item b) the maximum size of the interfering traffic is limited to *msrpLatencyMaxFrameSize* octets.

For item d, the propagation time, in the absence of better information, a value of 500 ns shall be used.

NOTE 2—This implies that no type of frame flow control can be used on the associated data stream packets.

The Listener can use this information to decide if the Latency is too large for acceptable presentation of the stream. The Accumulated Latency component is in units of nanoseconds.

35.2.2.8.7 FailureInformation

At the point when a Talker Advertise Declaration is transformed into a Talker Failed Declaration, the Bridge making the transformation adds information that indicates, to the Listeners registering the Talker Failed Declaration, the cause of the failure, and the identity of the Bridge and Bridge Port at which the failure occurred. The subcomponents of the FailureInformation include:

⁴⁵Example calculations for latency are contained in the paper “Calculating the Delay Added by Qav Stream Queue” [B4]

- a) The Bridge ID (13.24.1) of the Bridge that changed the Declaration Type from Advertise to Failed.
- b) The Reservation Failure Code which is represented by a single octet containing the value shown in Table 35-6.

Table 35-6—Reservation Failure Codes

Failure Code	Description of cause
1	Insufficient bandwidth
2	Insufficient Bridge resources
3	Insufficient bandwidth for traffic class.
4	StreamID in use by another Talker
5	Stream destination_address already in use
6	Stream preempted by higher rank
7	Reported latency has changed
8	Egress port is not AVB capable ¹
9	Use a different destination_address (i.e. MAC DA hash table full)
10	Out of MSRP resources
11	Out of MMRP resources
12	Cannot store destination_address (i.e. Bridge is out of MAC DA resources)
13	Requested priority is not an SR Class (3.175) priority
14	MaxFrameSize (35.2.2.8.4(a)) is too large for media
15	msrpMaxFanInPorts (35.2.1.4(f)) limit has been reached
16	Changes in FirstValue for a registered StreamID.
17	VLAN is blocked on this egress port (Registration Forbidden) ²
18	VLAN tagging is disabled on this egress port (untagged set)
19	SR class priority mismatch

¹A device could choose to use the asCapable variable from IEEE Std 802.1AS™-2011 [B8], 10.2.4.1, to help determine if its neighboring device is AVB capable. If the asCapable variable is FALSE for a particular port, then the neighboring device is not a time-aware system, and therefore not AVB capable.

²This Failure Code is never declared in a Talker Failed message since Talker attributes are not propagated on egress ports that have the associated VLAN blocked. The Bridge can still be queried by other means to learn why the Talker attribute was not declared.

35.2.2.9 MSRP FirstValue definitions (Domain discovery)

The Domain attribute contains all the information that a Bridge Port needs in order to determine the location of the SRP domain boundary (35.2.1.4(h)).

FirstValue shall be incremented one or more times when NumberOfValues is greater than one. Incrementing FirstValue for the MSRP Domain attribute is defined as follows:

- a) Add 1 to SRclassID (35.2.2.9.2), and
- b) Add 1 to SRclassPriority (35.2.2.9.3)

The choice of encoding and incrementing with this rule means that if one class (e.g. class B) is supported, then the FirstValue will be {5,2,VID} (class B, priority 2, and a VID) and the NumberOfValues field will be set to 1. If class A and class B are supported, with the default values, the FirstValue will again be {5,2,VID}, but the NumberOfValues fields will be set to 2. Applying the above incrementing rule to {5,2,VID} generates the value {6,3,VID}, i.e., class A, priority 3, and a VID, which is what we need for the default case.

35.2.2.9.1 Structure definition

The FirstValue for the Domain attribute in MSRPDUs exchanged according to the protocol specified in this subclause shall have the following structure:

- a) The first octet contains the *SRclassID* (35.2.2.9.2).
- b) Following the SRclassID is an octet containing the *SRclassPriority* (35.2.2.9.3).
- c) Following the SRclassPriority are two octets containing the *SRclassVID* (35.2.2.9.4).

The following BNF production gives the formal description of the MSRPDU FirstValue structure for the Domain attribute:

FirstValue ::= SRclassID, SRclassPriority, SRclassVID

Figure 35-3 illustrates the structure of the MSRPDU FirstValue components for the Domain attribute.

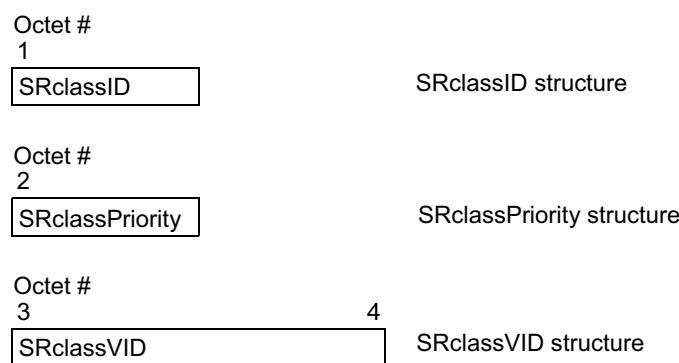


Figure 35-3—Format of the components of the Domain FirstValue

35.2.2.9.2 SRclassID

SRclassID is a numeric representation of the SR classes that are supported by a particular Bridge Port. The mapping for the first two SR classes is shown in Table 35-7.

Table 35-7—SR class ID

SR class	SR class ID
A	6
B	5

35.2.2.9.3 SRclassPriority

This field holds the Data Frame Priority (35.2.2.8.5(a)) value that will be used for streams that belong to the associated SR class. Whenever an end station's MAC_Operational (6.6.2) transitions to TRUE, its SRclassPriority shall be set to the default SR class priority (Table 6-6) until an SRclassPriority declaration is received from a neighboring device, at which time it shall be set to the declared value.

NOTE—This allows two end stations that are connected back-to-back to share streams, while also providing an end station the flexibility to change to the SRclassPriority of a attached network

35.2.2.9.4 SRclassVID

This field contains the SR_PVID (35.2.1.4(i)) that the associated streams will be tagged with by the Talker. Whenever an end station's MAC_Operational (6.6.2) transitions to TRUE, its SRclassVID shall be set to the default SR_PVID (Table 9-2) until an SRclassVID declaration is received from a neighboring device, at which time it shall be set to the declared value.

NOTE—This allows two end stations that are connected back-to-back to share streams, while also providing an end station the flexibility to change to the SRclassVID of a attached network

35.2.3 Provision and support of Stream registration service**35.2.3.1 Initiating MSRP registration and de-registration**

MSRP utilizes the following five declaration types (35.2.1.3) to communicate: Talker Advertise, Talker Failed, Listener Ready, Listener Ready Failed and Listener Asking Failed.

An SR Station behaving as a Talker will send a Talker Advertise declaration to inform the network about the characteristics (35.2.2.8) of the Stream it can provide. Bridges register this declaration, update some of the information contained within the Talker declaration, and forward it out the nonblocked (35.1.3.1) ports on the Bridge. If *talkerPruning* is enabled and the Bridge has the Stream's destination_address registered on one or more ports (via MMRP) it shall only forward the declarations out those ports. Eventually the Talker declaration will be registered by other SR stations.

If *talkerPruning* is enabled and the Stream's destination_address is not registered on any ports the Talker declaration shall not be forwarded.

An SR Station behaving as a Listener will receive the Talker Advertise declaration and register it. If the Listener is interested in receiving that Stream it will send a Listener Ready declaration back toward the Talker. The Bridge's MSRP MAP function will use the StreamID (35.2.2.8.2) to associate the Listener Ready with the Talker Advertise and forward the Listener Ready declaration only on the port that registered the Talker Advertise. This is referred to as Listener Pruning since the declarations are not forwarded out any other ports. The Bridge will also reserve the required bandwidth and configure its queues and the Filtering Database.

If any Bridge along the path from Talker to Listener does not have sufficient bandwidth or resources available its MSRP MAP function will change the Talker Advertise declaration to a Talker Failed declaration before forwarding it. Similarly a Listener Ready declaration will be changed to a Listener Asking Failed declaration if there is not sufficient bandwidth or resources available. This way both the Talker and Listener will know whether the reservation was successful or not.

In the case where there is a Talker attribute and Listener attribute(s) registered within a Bridge for a StreamID and a MAD_Leave.request is received for the Talker attribute, the Bridge shall act as a proxy for the Listener(s) and automatically generate a MAD_Leave.request back toward the Talker for those Listener attributes. This is a special case of the behavior described in 35.2.4.4.1.

Finally, there is the Listener Ready Failed declaration. This is used when there are two or more Listeners for a Stream. To simplify the explanation assume there are only two Listeners and one has sufficient bandwidth back to the Talker (signified by a Listener Ready), and the other does not (signified by a Listener Asking Failed). At some point in the network topology there will be a single Bridge that will receive the Listener Ready on one port from the first Listener and the Listener Asking Failed on another port from the second Listener. That Bridge's MSRP MAP function will merge (35.2.4.4.3) those two declarations into a single Listener Ready Failed declaration that will be forwarded to the Talker. When the Talker receives the Listener Ready Failed it will know there are one or more Listeners that want the Stream and can receive it, and there are one or more Listeners that want the Stream but have insufficient bandwidth or resources somewhere along the path to receive it. The Talker may still send the Stream, but it will realize that not all Listeners are going to receive it.

MSRP provides a set of Service Primitives that control the declarations of the attributes defined previously. The primitives associated with the Talker application entities are summarized in Table 35-8. Listener application entities have a corresponding set of primitives summarized in Table 35-9.

Table 35-8—Summary of Talker primitives

Name	Request	Indication
REGISTER_STREAM	35.2.3.1.1	—
Deregister_Stream	35.2.3.1.3	—
REGISTER_ATTACH	—	35.2.3.1.6
Deregister_Attach	—	35.2.3.1.8

Table 35-9—Summary of Listener primitives

Name	Request	Indication
REGISTER_STREAM	—	35.2.3.1.2
Deregister_Stream	—	35.2.3.1.4
REGISTER_ATTACH	35.2.3.1.5	—
Deregister_Attach	35.2.3.1.7	—

35.2.3.1.1 REGISTER_STREAM.request

A Talker application entity shall issue a REGISTER_STREAM.request to the MSRP Participant to initiate the advertisement of an available Stream.

A Talker may choose to enter a nonzero value in the Accumulated Latency that indicates the amount of latency in nanoseconds that a Stream will encounter before being passed to the MAC service interface.

On receipt of a REGISTER_STREAM.request the MSRP Participant shall issue a MAD_Join.request service primitive (10.2, 10.3). The attribute_type (10.2) parameter of the request shall carry the value of Talker Advertise Vector Attribute Type (35.2.2.4(a)) or Talker Failed Vector Attribute Type (35.2.2.4(b)), depending on the Declaration Type. The attribute_value (10.2) parameter shall carry the values from the REGISTER_STREAM.request primitive.

```
REGISTER_STREAM.request  (
    StreamID,
    Declaration Type,
    DataFrameParameters,
    TSpec,
    Data Frame Priority,
    Rank,
    Accumulated Latency,
    FailureInformation
)
```

35.2.3.1.2 REGISTER_STREAM.indication

A REGISTER_STREAM.indication notifies the Listener application entity that the referenced Stream is being advertised by a Talker somewhere on the attached network.

On receipt of a MAD_Join.indication service primitive (10.2, 10.3) with an attribute_type of Talker Advertise Vector Attribute Type or Talker Failed Vector Attribute Type the MSRP application shall issue a REGISTER_STREAM.indication to the Listener application entity. The REGISTER_STREAM.indication shall carry the values from the attribute_value parameter.

```
REGISTER_STREAM.indication  (
    StreamID,
    Declaration Type,
    DataFrameParameters,
    TSpec,
    Data Frame Priority,
    Rank,
    Accumulated Latency,
    FailureInformation
)
```

35.2.3.1.3 Deregister_STREAM.request

A Talker application entity shall issue a Deregister_STREAM.request to the MSRP Participant to remove the Talker's advertisement declaration, and thus remove the advertisement of a Stream, from the network.

On receipt of a Deregister_STREAM.request the MSRP Participant shall issue a MAD_Leave.request service primitive (10.2, 10.3) with the attribute_type set to the Declaration Type currently associated with

the StreamID. The attribute_value parameter shall carry the StreamID and other values that were in the associated REGISTER_STREAM.request primitive.

```
Deregister_Stream.request (  
    StreamID  
)
```

35.2.3.1.4 DREGISTER_STREAM.indication

A DREGISTER_STREAM.indication notifies the Listener application entity that the referenced Stream is no longer being advertised by a Talker.

On receipt of a MAD_Leave.indication service primitive (10.2, 10.3) with an attribute_type of Talker Advertise Vector Attribute Type or Talker Failed Vector Attribute Type the MSRP application shall issue a DREGISTER_STREAM.indication to the Listener application entity.

```
Deregister_Stream.indication (  
    StreamID  
)
```

35.2.3.1.5 REGISTER_ATTACH.request

A Listener application entity shall issue a REGISTER_ATTACH.request to the MSRP Participant to request attachment to the referenced Stream.

On receipt of a REGISTER_ATTACH.request the MSRP Participant shall issue a MAD_Join.request service primitive (10.2, 10.3). The attribute_type parameter of the request shall carry the value of Listener Vector Attribute Type (35.2.2.4(c)). The attribute_value shall contain the StreamID and the Declaration Type.

```
Register_Attach.request (  
    StreamID,  
    Declaration Type  
)
```

35.2.3.1.6 REGISTER_ATTACH.indication

A REGISTER_ATTACH.indication notifies the Talker application entity that the referenced Stream is being requested by one or more Listeners.

On receipt of a MAD_Join.indication service primitive (10.2, 10.3) with an attribute_type of Listener Vector Attribute Type the MSRP application shall issue a REGISTER_ATTACH.indication to the Talker application entity. The REGISTER_ATTACH.indication shall carry the values from the attribute_value parameter.

```
Register_Attach.indication (  
    StreamID,  
    Declaration Type  
)
```

35.2.3.1.7 DREGISTER_ATTACH.request

A Listener application entity shall issue a DREGISTER_ATTACH.request to the MSRP Participant to remove the request to attach to the referenced Stream.

On receipt of a Deregister_Attach.request the MSRP Participant shall issue a MAD_Leave.request service primitive (10.2, 10.3) with the attribute_type set to the Listener Vector Attribute Type. The attribute_value parameter shall carry the StreamID and the Declaration Type currently associated with the StreamID.

```
Deregister_Attach.request (
    StreamID
)
```

35.2.3.1.8 Deregister_Attach.indication

A Deregister_Attach.indication notifies the Talker application entity that the referenced Stream is no longer being requested by any Listeners.

On receipt of a MAD_Leave.indication service primitive (10.2, 10.3) with an attribute_type of Listener Vector Attribute Type the MSRP application shall issue a Deregister_Attach.indication to the Talker application entity. The Deregister_Attach.indication shall contain the StreamID.

```
Deregister_Attach.indication (
    StreamID
)
```

35.2.4 MSRP Attribute Propagation

There is no MSRP MAP function for Domain attributes. MSRP simply declares the characteristics of the SR classes that are supported on the Bridge Port regardless of what has been learned from Domain registrations on other Bridge Ports.

For the Talker and Listener attributes MSRP propagates attributes in a manner different from that described in 10.3 for MMRP and MVRP. In principle, the MAP performs MSRP Attribute Propagation when any of the following conditions occur:

- a) A MAD_Join.indication adds a new attribute to MAD (with the *new* parameter, 10.2, set to TRUE);
- b) A MAD_Leave.indication is issued by the MAD;
- c) An internal application declaration or withdrawal is made in a station;
- d) When the bandwidth of the underlying media changes (see *bandwidthAvailabilityChanged* notification in 34.3.2);
- e) A port becomes an SRP domain core port (3.177);
- f) If talkerPruning (35.2.4.3.1) is enabled and there is a change in the MMRP (10.9) attributes registered on a port.

The MSRP MAP function is responsible for adjusting and propagating Talker and Listener attributes throughout the bridged network. It also updates the Dynamic Reservation Entries (8.8.7) to specify which Streams shall be filtered and which shall be forwarded, along with updating the associated *streamAge* [35.2.1.4(c)]. The bandwidth associated with those Streams is reported to the queuing algorithms via the *operIdleSlope(N)* parameter [34.3(d)] on a per-port per-traffic class basis.

Streams of higher importance are given available bandwidth before streams of lower importance.

If insufficient bandwidth or resources are available the streams destination_address will be filtered and the failure will be noted in the Talker and Listener attributes declared from that Bridge.

A port shall only forward MSRP declarations for SR classes it supports. This will eliminate unnecessary priority remapping for traffic related to unsupported SR classes.

The following subclauses describe what the MSRP MAP function shall accomplish.

35.2.4.1 Stream importance

MSRP utilizes the stream Rank (35.2.2.8.5(b)) to decide which stream is more important than another.

In the case where two streams have the same Rank, the *streamAge* will be compared. If the Ranks are identical and the *streamAges* are identical the StreamIDs (35.2.2.8.2) will be compared and the numerically lower StreamID is considered to be more important.

35.2.4.2 Stream bandwidth calculations

As referenced in Table 35-5, the bandwidth requirements of a Stream include more than just the amount specified in the REGISTER_STREAM.request (35.2.3.1.1). SRP shall add the *perFrameOverhead* (34.4) associated with the media attached to the port.

If this port is on a Shared Media (35.1.1) the bandwidth requirements may need to be further increased. For example, a Stream transmitted from one station to another may have to be sent across the media twice. One time from the Talker station to the DMN, and a second time from the DMN to the Listener station. In this case the bandwidth requirements would need to be doubled.

The *totalFrameSize* for a stream on an outbound Port is therefore the sum of the following three amounts (doubled if each frame is transmitted on the media twice):

- a) MaxFrameSize (35.2.2.8.4(a));
- b) *perFrameOverhead* (34.4) associated with the media attached to the port;
- c) one (1) additional octet to account for slight differences (up to 200 ppm) in the class measurement interval between neighboring devices.

Multiply *totalFrameSize* by MaxIntervalFrames (35.2.2.8.4(b)) to arrive at the associated bandwidth (in bits per second), which is then used to update *operIdleSlope(N)* as shown in Table 35-13.

Streams that are in the Listener Ready or Listener Ready Failed state reduce the amount of bandwidth available to other Streams. Streams that have no Listeners, or the Listeners are in the Asking Failed state, do not reduce available bandwidth for other Streams since the Stream data will not be flowing through this outbound port. These details are considered when calculating how many Streams can flow through a particular port. The total amount of bandwidth available to a particular traffic class on a port is represented by *deltaBandwidth(N)* [34.3(b1)]. Subclause 34.3.1 describes the relationship of available bandwidth between traffic classes.

35.2.4.3 Talker attribute propagation

Table 35-10 describes the propagation of Talker attributes for a StreamID from one port of a Bridge to another. If no Talker attributes are registered for a StreamID then no Talker attributes for that StreamID will be declared on any other port of the Bridge. If a Talker Failed is registered then it will be propagated as a Talker Failed out all other nonblocked (35.1.3.1) ports on the Bridge.

Talker Advertise registrations require further processing by the MAP function. MAP will analyze available bandwidth and other factors to determine if the outbound port has enough resources available to support the Stream. MAP will also verify *msrpMaxFanInPorts*, if nonzero, will not be exceeded. If there are sufficient resources and bandwidth available MAP will declare a Talker Advertise. Otherwise, MAP will declare a Talker Failed on the outbound port and add appropriate FailureInformation.

Table 35-10—Talker Attribute Propagation per port

		Talker		
		(none)	Advertise	Failed
Listener	(none)	(none)	Talker Advertise or Talker Failed	Talker Failed
	Ready	(none)	Talker Advertise or Talker Failed	Talker Failed
	Ready Failed	(none)	Talker Advertise or Talker Failed	Talker Failed
	Asking Failed	(none)	Talker Advertise or Talker Failed	Talker Failed

35.2.4.3.1 Talker Pruning

By default Talker declarations are sent out all nonblocked ports. If *talkerPruning* is enabled and the destination_address (35.2.2.8.3(a)) of the Stream is found in the MAC Address Registration Entries (8.8.4) for the port, the declaration shall be forwarded. If the destination_address is not found the declaration shall be blocked and no Talker declaration of any type shall be forwarded.

35.2.4.4 Listener attribute propagation

Listener Attributes, unlike Talker Attributes, can be merged (35.2.4.4.3) from several Listeners on different ports into a single Listener declaration. There are two steps involved:

- 1) Processing of incoming Listener attribute registration on a port based upon the status of the associated Talker attribute registration,
- 2) Merging all the individual ports Listener attributes gathered in step 1, above, into a single Listener attribute to be declared on the nonblocked (35.1.3.1) outbound port which has the associated Talker registration. If no Talker attribute is registered within the Bridge for the StreamID associated with the Listener, the Listener attribute will not be propagated.

35.2.4.4.1 Incoming Listener attribute processing

Table 35-11 describes how Listener attributes are propagated from the incoming ports.

If no Listener attributes are registered on a particular port then no Listener attribute will be propagated to the Listener attribute merging (35.2.4.4.3) from that port.

If a Listener Asking Failed is registered on a port then it will be propagated as a Listener Asking Failed and merged with other Listener Attributes from other ports.

If no Talker attributes are associated with the Listener attribute, the Listener attribute will not be propagated.

If a Talker Failed is registered on another port for the associated Listener Ready or Listener Ready Failed then a Listener Asking Failed will be propagated and merged with other Listener attributes from other ports.

Table 35-11—Incoming Listener Attribute Propagation per port

		Talker		
		(none)	Advertise	Failed
Listener	(none)	(none)	(none)	(none)
	Ready	(none)	Listener Ready	Listener Asking Failed
	Ready Failed	(none)	Listener Ready Failed	Listener Asking Failed
	Asking Failed	(none)	Listener Asking Failed	Listener Asking Failed

If a Talker Advertise is registered on another port for the StreamID associated with a Listener Ready or Listener Ready Failed then that Listener attribute will be forwarded as-is and merged with Listener Attributes from other ports.

35.2.4.4.2 Updating Queuing and Forwarding information

When Incoming Listener attribute processing (35.2.4.4.1) has been completed for a port the Dynamic Reservation Entries (8.8.7) shall be updated as shown in Table 35-12.

Table 35-12—Updating Dynamic Reservation Entries

		Talker		
		(none)	Advertise	Failed
Listener	(none)	(no entry)	Filtering	Filtering
	Ready	Filtering	Forwarding	Filtering
	Ready Failed	Filtering	Forwarding	Filtering
	Asking Failed	Filtering	Filtering	Filtering

The *operIdleSlope(N)* [34.3(d)] shall be updated as shown in Table 35-13, dependant on the change to the Dynamic Reservation Entries. MSRP MAP processing can occur at any time (35.2.4), resulting in a change to which Streams are filtered and to which streams are forwarded. These changes in bandwidth requirements shall be reflected in the *operIdleSlope(N)* variable. Streams that have had their bandwidth removed shall decrease *operIdleSlope(N)*. Streams that have just been allocated bandwidth shall increase *operIdleSlope(N)*.

NOTE—The order of operations is important when updating the Dynamic Reservation Entries and the *operIdleSlope(N)* in order not to allow any associated streaming packets to be dropped. If the bandwidth utilization of a port is going to be increased (i.e. a Stream is going to be forwarded) the *operIdleSlope(N)* is updated before the Dynamic Reservation Entries. If the bandwidth utilization of a port is going to be decreased (i.e., a Stream is going to be filtered) the Dynamic Reservation Entries are updated before the *operIdleSlope(N)*.

Table 35-13—Updating *operIdleSlope(N)*

		Dynamic Reservation Entries prior to MSRP MAP running		
		(none)	Filtering	Forwarding
Dynamic Reservation Entries after MSRP MAP running	(none)	(no change)	(no change)	Decrease <i>operIdleSlope(N)</i>
	Filtering	(no change)	(no change)	Decrease <i>operIdleSlope(N)</i>
	Forwarding	Increase <i>operIdleSlope(N)</i>	Increase <i>operIdleSlope(N)</i>	(no change)

35.2.4.4.3 Merge Listener Declarations

Listener Registrations with the same StreamID shall be merged into a single Listener Declaration that will be declared on the port with the associated Talker registration. If no such Talker registration exists the Listener attribute is not declared. When this Declaration is sent:

- a) the Direction (35.2.1.2) is Listener;
- b) the Declaration Type (35.2.1.3) is determined according to Table 35-14;
- c) the StreamID (35.2.2.8.2) is that of the Listener Registrations.

Table 35-14—Listener Declaration Type Summation

First Declaration Type	Second Declaration Type	Resultant Declaration Type
Ready	none or Ready	Ready
	Ready Failed or Asking Failed	Ready Failed
Ready Failed	any	Ready Failed
Asking Failed	Ready or Ready Failed	Ready Failed
	none or Asking Failed	Asking Failed

35.2.4.5 MAP Context for MSRP

MSRPDUs can carry information about Streams in multiple VLANs, which in an MST environment, can be in different Spanning Tree Instances. Queue resources, however, are allocated and used according to priority parameters, not according to VLAN ID. Furthermore, on a shared medium, Streams can use the shared medium even on VLANs that are blocked on the Bridge's Port to that shared medium (e.g., consider an IEEE 802.11 AP that transmit packets between two stations that are on a VLAN that is blocked on the AP). Therefore there is a single context for MSRP attribute propagation that includes all Bridge Ports. The Declarations are filtered according to the state of the spanning tree, as described in 35.2.4.

All MSRPDUs sent and received by MSRP Participants in SST Bridges are transmitted as untagged frames.

35.2.5 Operational reporting and statistics

35.2.5.1 Dropped Stream Frame Counter

An implementation may support the ability to maintain a per-port per-traffic class count of data stream frames that are dropped for any reason. These are not MRP frames, but the data stream frames that flow between Talker and Listener(s) through the reservations established by MSRP.

35.2.6 Encoding

If an MSRP message is received from a Port with an event value (35.2.6) specifying the JoinIn or JoinMt message, and if the StreamID (35.2.2.8.2), and Direction (35.2.1.2) all match those of an attribute already registered on that Port, and the Attribute Type (35.2.2.4) or FourPackedEvent (35.2.2.7.2) has changed, then the Bridge should behave as though an **rLv!** event (with immediate leavetimer expiration in the Registrar state table) was generated for the MAD in the Received MSRP Attribute Declarations before the **rJoinIn!** or **rJoinMt!** event for the attribute in the received message is processed. This allows an Applicant to indicate a change in a stream reservation, e.g., a change from a Talker Failed to a Talker Advertise registration, without having to issue both a withdrawal of the old attribute, and a declaration of the new. A Listener attribute is also updated this way, for example, when changing from a Listener Ready to a Listener Ready Failed.

NOTE—This rule ensures that there is at most one Listener Declaration or one Talker Declaration for any given value of StreamID on any given port. In the unlikely situation where a Talker Advertise and a Talker Failed are received for the same Stream on the same port, the Talker Failed declaration takes precedence.

The following examples will help clarify the intent of this subclause. The first example describes the behavior when one attribute (Talker Advertise) is replaced by another (Talker Failed). The second example describes the behavior when the MSRP FourPackedEvents changes with a single Listener attribute.

This example illustrates processing of Talker Declaration changes. Assume a Bridge is receiving Talker Advertise declarations on an inbound Port. An emergency situation occurs and a 911 call is being placed via an entirely different Stream. The bandwidth that was available for the first Stream is now no longer available so the Bridge begins receiving Talker Failed declarations for that original stream. The MAP function realizes the Talker declaration has changed for the Stream and generates an internal leave event for the Talker Advertise and a join event for the Talker Failed. This behavior guarantees there will not be a declaration for a Talker Advertise and a Talker Failed for a single Stream existing within the Bridge at the same time.

As another example assume the same situation has occurred as described in the example above (a 911 call). For this scenario consider the Listener declarations flowing in the opposite direction. When there was bandwidth available the Bridge was declaring a Listener Ready, which was then changed to a Listener Asking Failed as soon as the 911 call came through. The other Bridge receiving these Listener declarations realized that the Listener attribute MSRP FourPackedEvents had changed and acted as if the Listener declaration had been withdrawn and replaced by the updated Listener declaration.

35.2.7 Attribute value support requirements

Implementations of MSRP shall maintain state information for all attribute values that support the Stream registrations (35.2.2.8).

Implementations of MSRP shall be capable of supporting any attribute value in the range of possible values that can be registered using Stream registrations (35.2.2.8); however, the maximum number of attribute values for which the implementation is able to maintain current state information is an implementation decision, and may be different for Talker attributes and Listener attributes. The number of values that the implementation can support shall be stated in the PICS.

Annex A

(normative)

PICS proforma—Bridge implementations⁴⁶

A.1 Introduction

The supplier of a protocol implementation that is claimed to conform to this standard shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use

- a) By the protocol implementer, as a checklist to reduce the risk of failure to conform to the standard through oversight;
- b) By the supplier and acquirer—or potential acquirer—of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma;
- c) By the user—or potential user—of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSs);
- d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A.2 Abbreviations and special symbols

A.2.1 Status symbols

M	mandatory
O	optional
O.n	optional, but support of at least one of the group of options labeled by the same numeral n is required
X	prohibited
pred:	conditional-item symbol, including predicate identification: see A.3.4
¬	logical negation, applied to a conditional item's predicate

A.2.2 General abbreviations

N/A	not applicable
PICS	Protocol Implementation Conformance Statement

⁴⁶Copyright 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.3 Instructions for completing the PICS proforma

A.3.1 General structure of the PICS proforma

The first part of the PICS proforma, implementation identification and protocol summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire, divided into several subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No) or by entering a value or a set or range of values. (Note that there are some items where two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered; the third column records the status of the item—whether support is mandatory, optional, or conditional: see also A.3.4. The fourth column contains the reference or references to the material that specifies the item in the main body of this standard, and the fifth column provides the space for the answers.

A supplier may also provide (or be required to provide) further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labeled A_i or X_i , respectively, for cross-referencing purposes, where i is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformance Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation’s configuration capabilities, in case that makes for easier and clearer presentation of the information.

A.3.2 Additional information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing on the answers to some items.

References to items of Additional Information may be entered next to any answer in the questionnaire and may be included in items of Exception Information.

A.3.3 Exception information

It may occasionally happen that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this item. Instead, the supplier shall write the missing answer into the Support column, together with an X_i reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to this standard.

NOTE—A possible reason for the situation described previously is that a defect in this standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional status

A.3.4.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply—mandatory or optional—are dependent on whether certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the “Not Applicable” answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form “**pred:** S” where **pred** is a predicate as described in A.3.4.2 below, and S is a status symbol, M or O.

If the value of the predicate is true (see A.3.4.2), the conditional item is applicable, and its status is indicated by the status symbol following the predicate: The answer column is to be marked in the usual way. If the value of the predicate is false, the “Not Applicable” (N/A) answer is to be marked.

A.3.4.2 Predicates

A predicate is one of the following:

- a) An item-reference for an item in the PICS proforma: The value of the predicate is true if the item is marked as supported and is false otherwise;
- b) A predicate-name, for a predicate defined as a boolean expression constructed by combining item-references using the boolean operator OR: The value of the predicate is true if one or more of the items is marked as supported;
- c) The logical negation symbol “ \neg ” prefixed to an item-reference or predicate-name: The value of the predicate is true if the value of the predicate formed by omitting the “ \neg ” symbol is false, and vice versa.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

A.3.4.3 References to the text of IEEE Std 802.1D-2004

Many tables in the PICS Proforma refer to the text of IEEE Std 802.1D (ANSI/IEEE Std 802.1D). A short form reference, of the form {D}X, is used in the “References” columns of these tables to denote references to clauses, subclauses, or tables in IEEE Std 802.1D, where X is the clause, subclause, or table identifier.

A.4 PICS proforma for IEEE Std 802.1Q—Bridge implementations

A.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification, e.g., name(s) and version(s) of machines and/or operating system names	

NOTE 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification.

NOTE 2—The terms “Name” and “Version” should be interpreted appropriately to correspond with a supplier’s terminology (e.g., Type, Series, Model).

A.4.2 Protocol summary, IEEE Std 802.1Q

Identification of protocol specification	IEEE Std 802.1Q-2011, IEEE Standards for Local and metropolitan area networks—MAC Bridges and Virtual Bridged Local Area Networks	
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	Amd. : Corr. : Amd. : Corr. :	
Have any Exception items been required? (See A.3.3: the answer “Yes” means that the implementation does not conform to IEEE Std 802.1Q)	No []	Yes []

Date of Statement	
-------------------	--

A.5 Major capabilities

Item	Feature	Status	References	Support
	If the implementation is an end station implementation, mark “N/A” and continue at Annex B.			N/A []
MAC	Do the implementations of MAC Technologies and support of the MAC Internal Sublayer Service conform to MAC standards as specified in 6.6 and 6.7? (If support of a specific MAC technology is claimed, any PICS Proforma(s) required by the standard specifying that technology shall also be completed.)	M	A.6 6.6, 6.7	Yes []
LLC	Is a class of LLC supporting Type 1 operations supported on all Bridge Ports in conformance with ISO/IEC 8802-2? (The PICS Proforma required by ISO/IEC 8802-2 shall also be completed.)	M	8.2, 8.3, 8.13, {D}7.2, {D}7.3, {D}7.12, ISO/IEC 8802-2	Yes []
RLY	Does the implementation relay and filter frames as specified?	M	8.5, 8.6, 8.7, 6.12, 8.8, A.7 {D}7.1, {D}7.5, {D}7.6, {D}7.7.	Yes []
BFS	Does the implementation maintain the information required to make frame filtering decisions and support Basic Filtering Services?	M	A.8 {D}7.1, {D}7.5, {D}7.8, {D}7.9.	Yes []
ADDR	Does the implementation conform to the provisions for addressing?	M	A.9, {D}7.12	Yes []
TPMR	Can the Bridge be configured to operate as a Two Port MAC Relay?	O.2	5.14	Yes [] No []
MSP	Is the operation of the MAC status propagation entity supported?	TPMR: M	23	Yes [] N/A []
RSTP	Is the Rapid Spanning Tree Protocol implemented?	¬TPMR:O.1 TPMR:X	5, 13, 14, A.10, {D}9	Yes [] No []
BPDU	Are transmitted BPDUs encoded and received BPDUs validated as specified?	¬TPMR:M TPMR:X	A.11, 13.27.26, 13.27.27, 13.27.28, 14, {D}9,	Yes [] No []
IMP	Are the required implementation parameters included in this completed PICS?	M	A.12 {D}7.9	Yes []
PERF	Are the required performance parameters included in this completed PICS? (Operation of the Bridge within the specified parameters shall not violate any of the other conformance provisions of this standard.)	M	A.13 {D}16	Yes []
MGT	Is management of the Bridge supported?	O PBBTE OR TPMR: M	A.14	Yes [] No []

A.5 Major capabilities (*continued*)

Item	Feature	Status	References	Support
RMGT	Is a remote management protocol supported?	MGT:O PBBTE OR TPMR: M	A.15 {D}5.2	Yes [] No []
MIB	Does the system implementation support management operations using SMIV2 MIB modules?	MGT:O	8.12, 17	Yes [] No []
TC	Are multiple traffic classes supported for relaying frames?	O	A.16 {D}7.7.3, {D}7.7.4.	Yes [] No []
EFS	Are Extended Filtering Services supported for relaying and filtering frames?	¬TPMR:O TPMR:X	A.17 6.12	Yes [] No []
MMRP	Is the operation of the Multiple MAC Registration Protocol (MMRP) supported?	EFS:M	5.4.1, A.20	Yes [] No []
VLAN	Does the implementation support the ability to insert tag headers into, modify tag headers in, and remove tag headers from relayed frames?	¬TPMR:M TPMR:X	5.4, 6.3, 6.8, 6.9, 8.6, 6.12, 9	Yes []
VTR	Does the implementation support a VID translation table?	¬TPMR:O TPMR:X	6.9	Yes [] No []
MVRP	Is automatic configuration and management of VLAN topology using MVRP supported?	¬TPMR:M TPMR:X	5.4, A.21	Yes []
MRP	Is the Multiple Attribute Registration Protocol (MRP) implemented in support of MRP Applications?	MMRP:M MVRP:M	10 A.20, A.21, A.22	Yes []
MRP1	Does the MRP implementation support operation of the Full Participant?	MRP:O.4	10 A.22	Yes [] No []
MRP2	Does the MRP implementation support operation of the Full Participant, point-to-point subset?	MRP:O.4	10 A.22	Yes [] No []
MSTP	Is the Multiple Spanning Tree protocol implemented?	¬TPMR:O.1 TPMR:X	5, 7, 8.4, 8.6.1, 8.8.8, 8.9, 8.10, 8.13.7, 11.2.3.1.2, 13, 14, A.18	Yes [] No []
VMGT	Does the implementation support VLAN management operations?	¬TPMR AND MGT:O TPMR:X	5.4.1, 12.10.2, 12.10.3	Yes [] No []
CB	Can the Bridge be configured to operate as a C-VLAN Bridge, recognizing and using C-TAGs?	O.2	5.9	Yes [] No []
PB	Can the Bridge be configured to operate as a Provider Bridge, recognizing and using S-TAGs?	O.2	5.10	Yes [] No []
PEB	Can the Bridge be configured to operate as a Provider Edge Bridge with one or more Ports operating as Customer Edge Ports?	PB:O	5.10.2	Yes [] No []

A.5 Major capabilities (*continued*)

Item	Feature	Status	References	Support
PB-2	State which Ports support the following values for the Provider Bridge Port Type: — Provider Network Port; — Customer Network Port; — Customer Edge Port.	PB:M	5.10	Ports: _____ Ports: _____ Ports: _____
CFM	Is Connectivity Fault Management implemented?	O PBBTE: M	5.4.1.4, 19, 20, 21, 22	Yes [] No []
CFM-T	Does the implementation support a CFM MD level zero MIP on each Port?	TPMR:M	5.13, 18–22	Yes []
BRG	Is this system a Bridge, and not a Station, for the purposes of Connectivity Fault Management?	CFM: O	22.4	Yes [] No []
BEB	Can the Bridge be configured to operate as a Backbone Edge Bridge, recognizing and using I-TAGs?	O.2	5.11	Yes [] No []
BEB-B	Can the Bridge be configured to operate as a Backbone Edge Bridge with one or more Ports operating as a Customer Backbone Ports?	BEB: O.3	5.11	Yes [] No []
BEB-I	Can the Bridge be configured to operate as a Backbone Edge Bridge with one or more Ports operating as a Provider Instance Port?	BEB: O.3	5.11	Yes [] No []
BEB-1	State which Ports support the following values for the Backbone Edge Bridge Port Type: —Provider Instance Port; —Customer Network Port; —Provider Network Port; —Customer Backbone Port.	BEB: M	5.11	PIP: _____ CNP: _____ PNP: _____ CBP: _____
DDCFM	Is management of data-driven and data-dependent connectivity faults implemented?	O	19, 29	Yes[] No []
PBBTE	Can the Bridge be configured by an external agent to provide TESIs?	O	8.4, 8.9, 25.10	Yes [] No []
EXAG	Is the active topology, learning and forwarding of the TESIs under the control of an external agent?	PBBTE: O.1	8.4, 8.9	Yes [] N/A []
ESPVID	Are the VIDs associated with ESPs, the ESP-VIDs, allocated to the TE-MSTID?	PBBTE: M	8.9	Yes [] N/A []
ESPFID	Is every ESP-VID allocated to a distinct FID?	PBBTE: M	25.10	Yes [] N/A []
BCBTE	Can the Bridge be configured to operate as a Backbone Core Bridge that provides TESIs?	PB AND PBBTE: O.5	5.10, 5.6.2	Yes [] N/A []
BEBTE	Can the Bridge be configured to operate as a Backbone Edge Bridge that provides TESIs?	BEB AND PBBTE: O.5	5.8.2, 5.11.1	Yes [] N/A []

A.5 Major capabilities (continued)

Item	Feature	Status	References	Support	
PS	Is protection switching supported?	BEBTE: M	5.8.2, 26.10.3	Yes []	N/A []
FQTSS	Does the implementation support forwarding and queuing for time-sensitive streams?	O	5.4.1.5, 34	Yes []	No []
CN	Is congestion notification implemented?	O	5.4.3, 30, 31, 32, 33	Yes []	No []
BRG1	Is this system a bridge, and not an end station, for the purposes of congestion notification?	CN: O		Yes []	No []
SRP	Does the implementation support the Stream Reservation Protocol?	O	35	Yes []	No []

A.6 Media Access Control methods

Item	Feature	Status	References	Support
	Which Media Access Control methods are implemented in conformance with the relevant MAC Standards?		5.4, 6.6, 6.7	
MAC-802.3	CSMA/CD, IEEE Std 802.3	O.2	6.7.1	Yes [] No []
MAC-802.11	Wireless LAN, IEEE Std 802.11	O.2	6.7.2	Yes [] No []
MAC-802.17	Resilient packet ring (RPR)	O.2	6.7.3	Yes [] No []
MAC-802.20-WB	IEEE Std. 802.20 Wideband Mode	O.2	6.7.4.1	Yes [] No []
MAC-802.20-625	IEEE Std. 802.20 625k-MC Mode	O.2	6.7.4.2	Yes [] No []
MAC-1	Has a PICS been completed for each of the Media Access Control methods implemented as required by the relevant MAC standards?	M		Yes[]
MAC-2	Do all the Media Access Control methods implemented support the MAC Internal Sublayer Service as specified.	M	6.6, 6.7, {D}6.4, {D}6.5	Yes []
MAC-3	Are the adminPointToPointMAC and operPointToPointMAC parameters implemented on all Ports?	M	6.6, 6.7, {D}6.4, {D}6.5	Yes []
MAC-4	Does the implementation support the use of the adminEdgePort and operEdgePort parameters on any Ports?	O	13.25.1, 13.25.30	Yes [] No []
MAC-4a	State which Bridge Ports support the adminEdgePort and operEdgePort parameters.			Ports _____
MAC-5	Is the priority of received frames set to the Default Priority where specified for the MAC?	M	6.7, {D}6.5.1, {D}6.5.4	Yes[]
MAC-6	Can the Default Priority be set for each Port?	O	6.7, {D}6.5.1, {D}6.5.4	Yes [] No []
MAC-7	Can the Default Priority be set to any of 0–7?	MAC-6:M	6.7, {D}6.5.1, {D}6.5.4	Yes[]

A.6 Media Access Control methods (*continued*)

Item	Feature	Status	References	Support
MAC-12	Is the minimum tagged frame length that can be transmitted on IEEE 802.3 Ports less than 68 (but 64 or more) octets?	MAC-802.3:O	6.7.1	Yes [] No [] N/A []
MAC-13	When transmitting untagged frames and the canonical_format_indicator parameter indicates that the mac_service_data_unit may contain embedded MAC Addresses in a format inappropriate to the destination MAC method, which of the following procedures is adopted by the Bridge: Convert any embedded MAC Addresses in the mac_service_data_unit to the format appropriate to the destination MAC method.	O.2	6.3, 6.6.1	Yes [] No []
MAC-14	Discard the frame without transmission on that Port.	O.2		Yes [] No []

A.7 Relay and filtering of frames

Item	Feature	Status	References	Support
RLY-1	Are received frames with MAC method errors discarded?	M	{D}6.4, 8.5	Yes []
RLY-2	Are user data frames the only type of frame relayed?	M	8.5	Yes []
RLY-3	Is the priority of each frame relayed regenerated as specified?	¬TPMR: M TPMR: O	6.7, 6.9.4, 8.5, 6.20	Yes [] N/A [] No []
RLY-4	Are the default values of the Priority Regeneration Table as specified for each Port?	M	6.9.4, Table 6-5	Yes []
RLY-5	Can the Priority Regeneration Table be modified?	O	6.9.4, Table 6-5	Yes []
RLY-6	Can the entries in the Priority Regeneration Table be set independently for each priority and Port and to any of the full range of values?	RLY-5: M	6.9.4, Table 6-5	Yes []
RLY-7	Are frames transmitted by an LLC User attached at a Bridge Port also submitted for relay?	¬TPMR: M TPMR: X	8.5	Yes [] No []
RLY-8	Are correctly received user data frames relayed subject to the conditions imposed by the Forwarding Process?	M	8.6, 8.6.2, 8.6.1, 8.6.3, 8.6.5, 8.6.4, 8.8, Table 8-7, Table 8-8, Table 8-9	Yes []
RLY-9	Is the order of relayed frames preserved as required by the forwarding process?	M	8.6.6	Yes []

A.7 Relay and filtering of frames (continued)

Item	Feature	Status	References	Support
RLY-10	Is a relayed frame submitted to a MAC Entity for transmission only once?	M	8.6.7	Yes []
RLY-11	Is a maximum bridge transit delay enforced for relayed frames?	M	8.6.7	Yes []
RLY-12	Are queued frames discarded if a Port leaves the Forwarding State?	M	8.6.7	Yes []
RLY-13	Is the default algorithm for selecting frames for transmission supported?	M	8.6.8	Yes []
RLY-14	Is the access priority of each transmitted frame as specified for each media access method?	M	6.5.9	Yes []
RLY-15	Is the FCS of frames relayed between Ports of the same MAC type preserved?	O	6.9	Yes [] No []
RLY-16	Is the undetected frame error rate greater than that achievable by preserving the FCS?	¬RLY-15:X	6.5.7, 6.9	No [] N/A []
RLY-17	Are CFM frames discarded if they enter the queue subsequent to the Port leaving the Forwarding state?	CFM: X	8.6.7	No [] N/A []
RLY-18	Are CFM frames discarded when the Port leaves the Forwarding state?	CFM: O	8.6.7	Yes [] No []
RLY-19	Are received frames discarded (or not discarded) in accordance with the specification of a Provider Instance Port, Customer Backbone Port, or Backbone Service Instance Multiplex Entity?	BEB: M	6.10.1, 6.11.1, 6.18.1	Yes [] No []

A.8 Basic Filtering Services

Item	Feature	Status	References	Support
BFS-1	Are correctly received user data frames submitted to the Learning Process if the learning variable is set??	¬TPMR: M TPMR: X	8.6, 8.7, 5.13	Yes [] No []
BFS-2	Are correctly received frames of types other than user data frames submitted to the Learning Process?	¬TPMR: O TPMR: X	8.6, 8.7, 5.13	Yes [] N/A [] No []
BFS-3	Does the Filtering Database support creation and update of Dynamic Filtering Entries by the Learning Process?	¬TPMR: M TPMR: X	8.7, 8.8, 8.8.3, 5.13	Yes [] No []
BFS-4	Are Dynamic Filtering Entries created and updated if and only if the Port State permits?	M	8.7, 8.8.3	Yes []
BFS-5	Are Dynamic Filtering Entries created on receipt of frames with a group source address?	X	8.7, 8.8.3	No []

A.8 Basic Filtering Services (*continued*)

Item	Feature	Status	References	Support
BFS-6	Can a Dynamic Filtering Entry be created that conflicts with an existing Static Filtering Entry?	X	8.7, 8.8, 8.8.2, 8.8.3	No []
BFS-7	Are existing Dynamic Filtering Entries removed to allow creation of a new entry if the Filtering Database is full?	¬TPMR: O TPMR: X	8.7, 8.8.3, 5.13	Yes [] N/A [] No []
BFS-8	Does the Filtering Database contain Static Filtering Entries?	M	8.8.2	Yes []
BFS-9	Are Static Filtering Entries aged out?	X	8.8	No []
BFS-10	Can Static Filtering Entries be created, modified, and deleted by management?	¬(TPMR OR PBBTE): O PBBTE: M TPMR: X	8.8, 5.13	N/A [] Yes [] No []
BFS-11	Can Static Filtering Entries be made for individual and Group MAC Addresses?	BFS-10:M	8.8.2	Yes [] N/A[]
BFS-12	Can a Static Filtering Entry be made for the broadcast MAC Address?	BFS-10:M	8.8.2	Yes [] N/A[]
BFS-13	Can a Static Filtering Entry specify a forwarding Port Map?	BFS-10:M	8.8.2	Yes [] N/A[]
BFS-14	Can a Static Filtering Entry specify a filtering Port Map?	BFS-10:M	8.8.2	Yes [] N/A[]
BFS-15	Does the creation of a Static Filtering Entry remove any conflicting information in a Dynamic Filtering Entry for the same address?	M	8.8.2, 8.8.3	Yes []
BFS-16	Can a separate Static Filtering Entry with a Port Map be created for each inbound Port?	¬TPMR: O TPMR: X	8.8.2, 5.13	Yes [] N/A [] No []
BFS-17	Are Dynamic Filtering Entries aged out of the Filtering Database if not updated?	M	8.8.3	Yes []
BFS-18	Can more than one Dynamic Filtering Entry be created for the same MAC Address?	X	8.8.3	No []
BFS-19	Can the Bridge be configured to use the recommended default Ageing Time?	¬TPMR: O TPMR: X	8.8.3, Table 8-6 5.13	Yes [] N/A [] No []
BFS-20	Can the Bridge be configured to use any value in the range specified for Ageing Time?	¬TPMR: O TPMR: X	8.8.3, Table 8-6 5.13	Yes [] N/A [] No []
BFS-21	Is the Filtering Database initialized with the entries contained in the Permanent Database?	M	8.8.11	Yes []
BFS-22	Are correctly received user data frames submitted to the Learning Process if they are associated with an ESP-VID?	PBBTE: X	8.4, 8.6.1	Yes [] No [] N/A []
BFS-23	Are correctly received user data frames that are associated with an ESP-VID discarded if their destination MAC address is unknown?	PBBTE: M	8.8.1	Yes [] No [] N/A []

A.9 Addressing

Item	Feature	Status	References	Support
ADDR-1	Does each Port have a separate MAC Address?	M	8.13.2	Yes []
ADDR-2	Are frames addressed to a MAC Address for a Port and received from or relayed to the attached LAN submitted to LLC Service User for the destination LLC Address?	M	8.5, 8.13.2	Yes []
ADDR-3	Are all BPDUs and MRP PDUs transmitted using the Bridge Spanning Tree Protocol LLC Address?	M	8.13.3, Table 8-12	Yes []
ADDR-4	Are PDUs addressed to the Bridge Spanning Tree Protocol Address with an unknown Protocol Identifier discarded on receipt?	M	14.4, {D}9.3.4	Yes []
ADDR-5	Are all BPDUs generated by a Spanning Tree Protocol Entity associated with a C-VLAN component transmitted to the Bridge Group Address?	CB OR PEB: M	8.13.3, Table 8-1, Table 8-2	Yes [] N/A []
ADDR-6	Are all BPDUs generated by a Spanning Tree Protocol Entity associated with an S-VLAN component, I-component, or B-component transmitted to the Provider Bridge Group Address?	PB: M BEB:M	Table 8-1	Yes [] N/A []
ADDR-7	Are all MRPDUs transmitted to the Group Address assigned for the MRP Application?	M	8.13.3, {D}Table 12-1 Table 8-1	Yes []
ADDR-8	Is it possible to create entries in the Permanent or Filtering Databases for unsupported MRP application addresses or delete or modify entries for supported application addresses?	X	8.13.3	No []
ADDR-9	Is the source MAC address of BPDUs and MRPDUs for MRP Applications supported by the Bridge address of the transmitting Port?	M	8.13.3	Yes []
ADDR-10	Is the source MAC address of BPDUs generated by a Spanning Tree Entity associated with a C-VLAN component of a Provider Edge Bridge the address of the associated Customer Edge Port?	PEB:M	12.13	Yes [] N/A []
ADDR-11	Is Bridge Management accessible through a Port using the MAC Address of the Port?	MGT:O	8.13.7	Yes [] No []
ADDR-12	Is an EUI-48 Universally Administered MAC Address assigned to each Bridge as its Bridge Address?	M	8.13.8	Yes []
ADDR-13	Is the Bridge Address the Address of a Port?	O	8.13.8	Yes [] No []

A.9 Addressing (*continued*)

Item	Feature	Status	References	Support
ADDR-14	Is the Bridge Address the Address of Port 1?	ADDR-13: O	8.13.8	Yes [] No []
ADDR-15	Are frames addressed to any of the C-VLAN component Reserved Addresses relayed by a C-VLAN component?	CB or PB:X	8.13.4, Table 8-1	No [] N/A []
ADDR-16	Are frames addressed to any of the S-VLAN component Reserved Addresses relayed by an S-VLAN component, I-component, or B-component?	PB:X BEB:X	Table 8-2	No [] N/A []
ADDR-17	Is it possible to delete or modify entries in the Permanent and Filtering Databases for the Reserved Addresses?	X	8.13.4, {D}7.12.6	No []
ADDR-18	Are PIPs capable of filtering frames received with a B-SA address matching the PIP MAC address?	BEB-I: O	5.7.1	Yes [] N/A [] No []
ADDR-19	Is the address of an internal PIP on Backbone Edge Bridge supporting a TESI configured to take the value of the MAC address of the connected CBP?	BEBTE:M	25.10	Yes [] N/A []

A.10 Rapid Spanning Tree Protocol

Item	Feature	Status	References	Support
	If item RSTP is not supported, mark “N/A” and continue at A.11			N/A[]
RSTP-1	Does each Bridge have a unique identifier based on the Bridge Address, and a unique identifier for each Port?	RSTP:M	13.2	Yes []
RSTP-2	Can each Port be configured as an edge port by setting the adminEdgePort parameter?	RSTP:O	13.4	Yes [] No []
RSTP-3	Can each Port be configured to automatically determine if it is an edge port by setting the autoEdgePort parameter?	RSTP:O	13.4	Yes [] No []
RSTP-4	Are learned MAC addresses transferred from a retiring Root Port to a new Root Port?	RSTP:O	13.17	Yes [] No []
RSTP-5	Is the Spanning Tree Protocol Entity reinitialized if the Force Protocol Version parameter is modified?	RSTP:M	13.24	Yes []
RSTP-6	Are spanning tree priority vectors and Port Role assignments recomputed if the Bridge Identifier Priority, Port Identifier Priority, or Port Path Costs change?	RSTP:M	13.24	Yes []
RSTP-7	Is the txCount variable for a Port set to zero if the Port’s Transmit Hold Count is modified?	RSTP:M	13.24	Yes []

A.10 Rapid Spanning Tree Protocol (*continued*)

Item	Feature	Status	References	Support
RSTP-8	Are the recommended default values of Migrate Time, Bridge Hello Time, Bridge Max Age, Bridge Forward Delay, and Transmit Hold Count used?	RSTP:O	13.23	Yes [] No []
RSTP-9	Can the Bridge Max Age, Bridge Forward Delay, and Transmit Hold Count parameters be set?	RSTP:O	13.23, 13.24	Yes [] No []
RSTP-10	Can Bridge Max Age, Bridge Forward Delay, Transmit Hold Count be set to any value in the permitted range?	RSTP-9: M	13.23, 13.28, Table 13-5	Yes [] N/A[]
RSTP-11	Are the relationships between Bridge Hello Time, Bridge Max Age, and Bridge Forward Delay enforced?	RSTP-9: M	13.23	Yes [] N/A[]
RSTP-12	Are the recommended values of Bridge Identifier Priority, Port Path Costs, and Port Identifier Priorities used?	RSTP:O	13.16	Yes [] No []
RSTP-13	Can the Bridge Identifier Priority, Port Path Costs, and Port Identifier Priorities be set?	RSTP:O	13.1, 13.4.1, 13.16, 13.23, 13.24, 13.25	Yes [] No []
RSTP-14	Can the Bridge Identifier Priority and Port Identifier Priorities be set to any of the values in the ranges specified?	RSTP-13: M	13.16	Yes [] N/A[]
RSTP-15	Can the Port Path Cost for each Port be set to any of the values in the specified range?	RSTP-13: M	13.16	Yes [] N/A[]
RSTP-16	Are Port Path Costs changed automatically by default if port speeds change?	RSTP:X	13.16	No []
RSTP-17	Is one instance of the Port Role Selection state machine implemented for the Bridge; is one instance of each of the Port Timers, Port Receive, Port Protocol Migration, Bridge Detection, Port Transmit, Port Information, Port Role Transition, Port State Transition, and Topology Change state machines implemented per Port; and are the referenced definitions and declarations followed for all machines?	RSTP:M	13.5.2, 13.22, 13.34, 13.28, 13.29, 13.30, 13.31, 13.32, 13.33, 13.35, 13.36, 13.37	Yes []
RSTP-18	Is it possible to set each Port Protocol Migration state machine's mcheck variable?	RSTP:O	13.25.25	Yes [] No []
RSTP-19	Is a single instance of each of the timer variables implemented per Port?	RSTP:M	13.28	Yes []
RSTP-20	Are the values for maximum RSTP processing delay and maximum BPDU transmission delay ever exceeded for any of the specified external events, actions, internal events, or transmissions?	RSTP:X	{D}17.32, {D}Table 17-5	No []
RSTP-21	Does each C-VLAN component of a Provider Edge Bridge operate an instance of Rapid Spanning Tree as modified for Customer Edge Ports?	RSTP AND PEB:M	13.39	Yes []

A.10 Rapid Spanning Tree Protocol (*continued*)

Item	Feature	Status	References	Support
RSTP-22	Does each I-component of a Backbone Edge Bridge operate an instance of Rapid Spanning Tree as modified for Virtual Instance Ports?	RSTP AND BEB:M	13.40	Yes []
RSTP-23	Can each Port be configured to support detection of fragile bridges by setting the autoIsolatePort parameter?	RSTP:O	13.21, 13.25.6	Yes [] No []
RSTP-24	Can a port be configured as an L2 Gateway Port by setting the isL2gp parameter?	RSTP:O	13.18, 13.25.19	Yes [] No []
RSTP-25	Can the pseudoRootId parameter be set for an L2 Gateway Port?	RSTP-24:M	13.18, 13.25.37	Yes [] N/A[]
RSTP-26	Is one instance of the L2 Gateway Port state machine implemented for each L2GP port?	RSTP-24:M	13.38	Yes [] N/A[]

A.11 BPDU encoding

Item	Feature	Status	References	Support
BPDU-1	Do all BPDUs contain an integral number of octets?	M	14, {D}9.1.1	Yes []
BPDU-2	Are all the following BPDU parameter types encoded as specified? Protocol Identifiers Protocol Version Identifiers BPDU Types Flags Bridge Identifiers Root Path Cost Port Identifiers Timer Values	M	14, {D}9.1.1, {D}9.2 14, {D}9.2.1 14, {D}9.2.2 14, {D}9.2.3 14, {D}9.2.4 14, {D}9.2.5 14, {D}9.2.6 14, {D}9.2.7 14, {D}9.2.8	Yes []
BPDU-3	Do Configuration BPDUs have the format, parameters, and parameter values specified?	M	14, {D}9.3.1, 14, {D}17.21.19	Yes []
BPDU-4	Do Topology Change Notification BPDUs have the format, parameters, and parameter values specified?	M	14, {D}9.3.2, {D}17.21.21	Yes []
BPDU-5	Do Rapid Spanning Tree BPDUs have the format, parameters, and parameter values specified?	M	14, {D}9.3.3, {D}17.21.20	Yes []
BPDU-6	Are received BPDUs validated as and only as specified?	M	14, {D}9.3.4	Yes []
BPDU-7	Does the implementation process BPDUs of prior and possible later protocol versions as specified?	M	14, {D}9.3.4	Yes []
BPDU-8	Do Multiple Spanning Tree BPDUs have the format, parameters, and parameter values specified?	MSTP:M	14.3	Yes []

A.12 Implementation parameters

Item	Feature	Status	References	Support
IMP-1	State the Filtering Database Size.	M	8.8	_____ entries
IMP-2	State the Permanent Database Size.	M	8.8	_____ entries
IMP-3	State the maximum number of VIDs supported by the implementation.	M	5.4, 8.8	_____ VIDs
IMP-4	State the range of VID values supported by the implementation.	M	5.4, 8.8	0 through _____
IMP-5	State the maximum number of FIDs that can be supported by the implementation.	M	8.8.8	_____ FIDs
IMP-6	State the maximum number of VIDs that can be allocated to each FID.	M	8.8.8	_____ VIDs
IMP-7	State the number of VLAN Learning Constraints that can be configured in the implementation.	MGT-56:M	5.4.1, 8.8.8, 12.10.3	_____ Constraints
IMP-8	State the number of MSTIs supported.	MSTP:M	5.4.1, 8.8.8, 13.13	_____ MSTIs N/A []

A.13 Performance

Item	Feature	Status	References	Support
PERF-1	Specify a Guaranteed Port Filtering Rate, and the associated measurement interval TF , for each Bridge Port in the format specified below.	M	{D}16.1	
PERF-2	Specify a Guaranteed Bridge Relaying Rate, and the associated measurement interval TR , in the format specified below. Supplementary information shall clearly identify the Ports.	M	{D}16.2	

Guaranteed Bridge Relaying Rate	TR
----- frames per second	----- second(s)

Port number(s) or other identification	Guaranteed port filtering rate (specify for all ports)	TF (specify for all ports)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)
	----- frames per second	----- second(s)

A.14 Bridge management

Item	Feature	Status	References	Support
	If items MGT or TPMR are not supported, mark N/A and continue at A.15.			N/A []
MGT-1	Discover Bridge	MGT:M	12.4.1.1	Yes []
MGT-2	Read Bridge	MGT:M	12.4.1.2	Yes []
MGT-3	Set Bridge Name	MGT:M	12.4.1.3	Yes []
MGT-4	Reset Bridge	MGT:M	12.4.1.4	Yes []
MGT-5	Read Port	MGT:M	12.4.2.1	Yes []
MGT-6	Set Port Name	MGT:M	12.4.2.2	Yes []
MGT-7	Read Forwarding Port Counters	MGT:M	12.6.1.1	Yes []
MGT-8	Read Port Default Priority	MGT:M	12.6.2.1	Yes [] N/A []
MGT-9	Set Port Default Priority	MGT AND MAC-6:M	12.6.2.2	Yes [] N/A []
MGT-10	Read Port Priority Regeneration Table	MGT AND RLY-5:M	12.6.2.3	Yes [] N/A []
MGT-11	Set Port Priority Regeneration Table	MGT AND RLY-5:M	12.6.2.4	Yes [] N/A []
MGT-12	Read Port Traffic Class Table	MGT AND TC:M	12.6.3.1	Yes [] N/A []
MGT-13	Set Port Traffic Class Table	MGT AND TC-3:M	12.6.3.2	Yes [] N/A []
MGT-14	Read Port Priority Code Point Selection	MGT and VLAN-29:M	12.6.2.5	Yes [] N/A []
MGT-15	Set Port Priority Code Point Selection	MGT AND VLAN-29:M	12.6.2.6	Yes [] N/A []
MGT-16	Read Priority Code Point Decoding Table	MGT AND VLAN-29:M	12.6.2.7	Yes [] N/A []
MGT-17	Set Priority Code Point Decoding Table	MGT AND VLAN-29:O	12.6.2.8	Yes [] No [] N/A []
MGT-18	Read Priority Code Point Encoding Table	MGT AND VLAN-29:M	12.6.2.9	Yes [] N/A []
MGT-19	Set Priority Code Point Encoding Table	MGT AND VLAN-29:O	12.6.2.10	Yes [] No [] N/A []
MGT-20	Read Use_DEI parameter	MGT:M	12.6.2.12	Yes [] N/A []
MGT-21	Set Use_DEI parameter	MGT:O	12.6.2.13	Yes [] No [] N/A []
MGT-22	Read Require Drop Encoding parameter	MGT AND VLAN-29:M	12.6.2.14	Yes [] N/A []
MGT-23	Set Require Drop Encoding parameter	MGT AND VLAN-29:M	12.6.2.15	Yes [] N/A []
MGT-24	Read Service Access Priority Selection	MGT AND VLAN-30:M	12.6.2.16	Yes [] N/A []
MGT-25	Set Service Access Priority Selection	MGT AND VLAN-30:M	12.6.2.17	Yes [] No [] N/A []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support	
MGT-26	Read Service Access Priority Table	MGT AND VLAN-30:M	12.6.2.18	Yes []	N/A []
MGT-27	Set Service Access Priority Table	MGT AND VLAN-30:O	12.6.2.19	Yes []	No [] N/A []
MGT-28	Read Filtering Database	MGT:M	12.7.1.1	Yes []	
MGT-29	Set Filtering Database Ageing Time	MGT:M	12.7.1.2	Yes []	
MGT-30	Read Permanent Database	MGT:M	12.7.6.1	Yes []	
MGT-31	Create Filtering Entry	MGT:M	12.7.7.1	Yes []	
MGT-32	Delete Filtering Entry	MGT:M	12.7.7.2	Yes []	
MGT-33	Read Filtering Entry	MGT:M	12.7.7.3	Yes []	
MGT-34	Read Filtering Entry Range	MGT:M	12.7.7.4	Yes []	
MGT-35	Read CIST Bridge Protocol Parameters	MGT:M	12.8.1.1	Yes []	
MGT-36	Set CIST Bridge Protocol Parameters	MGT:M	12.8.1.3	Yes []	
MGT-37	Read CIST Port Parameters	MGT:M	12.8.2.1	Yes []	
MGT-38	Set CIST Port Parameters	MGT:M	12.8.2.3	Yes []	
MGT-39	Force BPDU Migration Check	MGT:M	12.8.2.5	Yes []	
MGT-40	Read MRP Timers	MGT AND MRP:M	12.9.1.1	Yes []	N/A []
MGT-41	Set MRP Timers	MGT AND MRP:M	12.9.1.2	Yes []	N/A []
MGT-42	Read MRP Protocol Controls	MGT AND MRP:M	12.9.2.1	Yes []	N/A []
MGT-43	Set MRP Protocol Controls	MGT AND MRP:M	12.9.2.2	Yes []	N/A []
MGT-44	Read MSTI Bridge Protocol Parameters	MGT AND MSTP:M	12.8.1.2	Yes []	N/A []
MGT-45	Set MSTI Bridge Protocol Parameters	MGT AND MSTP:M	12.8.1.4	Yes []	N/A []
MGT-46	Read MSTI Port Parameters	MGT AND MSTP:M	12.8.2.2	Yes []	N/A []
MGT-47	Set MSTI Port Parameters	MGT AND MSTP:M	12.8.2.4	Yes []	N/A []
MGT-48	Read Bridge VLAN Configuration	MGT:M	12.10.1.1	Yes []	N/A []
MGT-49	Configure PVID values	MGT:M	12.10.1.2	Yes []	N/A []
MGT-50	Configure Acceptable Frame Types parameter	MGT AND VLAN-2:M	12.10.1.3	Yes []	N/A []
MGT-51	Configure Enable Ingress Filtering parameters	MGT AND VLAN-9:M	12.10.1.4	Yes []	N/A []
MGT-52	Reset Bridge	MGT:M	12.10.1.5	Yes []	N/A []
MGT-53	Notify VLAN Registration Failure	MGT:M	12.10.1.6	Yes []	N/A []
MGT-54	Read VLAN Configuration	MGT:M	12.10.2.1	Yes []	N/A []
MGT-55	Create VLAN Configuration	MGT:M	12.10.2.2	Yes []	N/A []
MGT-56	Delete VLAN Configuration	MGT:M	12.10.2.3	Yes []	N/A []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
	If item MSTP is not supported, mark N/A and continue at MGT-68			N/A []
MGT-57	Read MSTI List	MGT AND MSTP:M	12.12.1.1	Yes []
MGT-58	Create MSTI	MGT AND MSTP:M	12.12.1.2	Yes []
MGT-59	Delete MSTI	MGT AND MSTP:M	12.12.1.3	Yes []
MGT-60	Read FID to MSTI allocation	MGT AND MSTP:M	12.12.2.1	Yes []
MGT-61	Set FID to MSTI allocation	MGT AND MSTP:M	12.12.2.2	Yes []
MGT-62	Read MST Configuration Table Element	MGT AND MSTP:M	12.12.3.1	Yes []
MGT-63	Read VIDs assigned to MSTID	MGT AND MSTP:M	12.12.3.2	Yes []
MGT-64	Read MSTI Configuration Identifier	MGT AND MSTP:M	12.12.3.3	Yes []
MGT-65	Set MSTI Configuration Identifier	MGT AND MSTP:M	12.12.3.4	Yes []
MGT-66	Does the implementation support the configuration of VLAN learning constraints via management?	MGT:O	5.4.1, 8.8.8, 12.10.3	Yes [] No []
MGT-67	Read VLAN Learning Constraints	MGT-56:M	12.10.3.1	Yes [] N/A []
MGT-68	Read VLAN Learning Constraints for VID	MGT-56:M	12.10.3.2	Yes [] N/A []
MGT-69	Set VLAN Learning Constraint	MGT-56:M	12.10.3.3	Yes [] N/A []
MGT-70	Delete VLAN Learning Constraint	MGT-56:M	12.10.3.4	Yes [] N/A []
MGT-71	Notify Learning Constraint Violation	MGT-56:M	12.11	Yes [] N/A []
MGT-72	Does the implementation support configuration of VID to FID allocations via management?	MGT:O	5.4.1, 8.8.8.1, 12.10.3	Yes [] No [] N/A []
MGT-73	Read VID to FID allocations	MGT-62:M	12.10.3.5	Yes [] N/A []
MGT-74	Read FID allocation for VID	MGT-62:M	12.10.3.6	Yes [] N/A []
MGT-75	Read VIDs allocated to FID	MGT-62:M	12.10.3.7	Yes [] N/A []
MGT-76	Set VID to FID allocation	MGT-62:M	12.10.3.8	Yes [] N/A []
MGT-77	Delete VID to FID allocation	MGT-62:M	12.10.3.9	Yes [] N/A []
MGT-78	Support Bridge management for the bridge protocol entity in all supported spanning trees	MGT AND MSTP:M	5.4, 12.8	Yes []
MGT-79	Support independent management of bridge and port priority and path cost per spanning tree	MGT AND MSTP:M	5.4, 12.8.1	Yes []
MGT-80	Support VLAN management per spanning tree	MGT AND MSTP:M	5.4, 12.10.1	Yes []
MGT-81	Support MSTI configuration management	MGT AND MSTP:M	5.4, 12.12	Yes []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-82	Read Provider Bridge Port Type	MGT AND PB:M MGT AND BEB:M	12.13.1.1	Yes [] N/A []
MGT-83	Configure Provider Bridge Port Type	MGT AND PB:O MGT AND BEB:O	12.13.1.2	Yes [] N/A [] No []
MGT-84	Read VID Translation Table Entry	MGT AND VLAN-31:M	12.13.2.1	Yes [] N/A []
MGT-85	Configure VID Translation Table Entry	MGT AND VLAN-31:M	12.13.2.2	Yes [] N/A []
MGT-86	Read C-VID Registration Table Entry	MGT AND PEB:M	12.13.3.1	Yes [] N/A []
MGT-87	Configure C-VID Registration Table Entry	MGT AND PEB:M	12.13.3.2	Yes [] N/A []
MGT-88	Read Provider Edge Port Configuration	MGT AND PEB:M	12.13.3.3	Yes [] N/A []
MGT-89	Set Provider Edge Port Configuration	MGT AND PEB:M	12.13.3.4	Yes [] N/A []
MGT-90	Read Service Priority Regeneration Table	MGT AND PEB:M	12.13.3.5	Yes [] N/A []
MGT-91	Set Service Priority Regeneration Table	MGT AND PEB:O	12.13.3.6	Yes [] No [] N/A []
MGT-92	Does the Bridge provide control of all of the required CFM managed objects?	BRG AND CFM: M	item h) in 5.4.1.4, 12.14	Yes []
MGT-93	Reserved			
	Questions MGT-94 through MGT-119 refer to operations related to CFM managed objects.			
MGT-94	Read Maintenance Domain list	CFM: M	12.14.1.1	Yes []
MGT-95	Create Maintenance Domain managed object	CFM: M	12.14.1.2	Yes []
MGT-96	Delete Maintenance Domain managed object	CFM: M	12.14.1.3	Yes []
MGT-97	Read CFM Stack managed object	CFM: M	12.14.2.1	Yes []
MGT-98	Read Default MD Level managed object	CFM: M	12.14.3.1	Yes []
MGT-99	Write Default MD Level managed object	CFM: M	12.14.3.2	Yes []
MGT-100	Read Configuration Error List managed object	CFM: M	12.14.4.1	Yes []
MGT-101	Read Maintenance Domain managed object	CFM: M	12.14.5.1	Yes []
MGT-102	Write Maintenance Domain managed object	CFM: M	12.14.5.2	Yes []
MGT-103	Create Maintenance Association managed object	CFM: M	12.14.5.3	Yes []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-104	Delete Maintenance Association managed object	CFM: M	12.14.5.4	Yes []
MGT-105	Read Maintenance Association managed object	CFM: M	12.14.6.1	Yes []
MGT-106	Write Maintenance Association managed object	CFM: M	12.14.6.2	Yes []
MGT-107	Create Maintenance association End Point managed object	CFM: M	12.14.6.3	Yes []
MGT-108	Delete Maintenance association End Point managed object	CFM: M	12.14.6.4	Yes []
MGT-109	Read Maintenance association End Point managed object	CFM: M	12.14.7.1	Yes []
MGT-110	Total number of out-of-sequence CCMs received	CFM: O	item v) in 12.14.7.1.3, 20.16.12	Yes [] No []
MGT-111	Total number of LBRs received with data match errors	CFM: O	item aa) in 12.14.7.1.3	Yes [] No []
MGT-112	Write Maintenance association End Point managed object	CFM: M	12.14.7.2	Yes []
MGT-113	Transmit Loopback Messages	CFM: M	12.14.7.3	Yes []
MGT-114	Transmit Linktrace Message	CFM: M	12.14.7.4	Yes []
MGT-115	Read Linktrace Reply	CFM: M	12.14.7.5	Yes []
MGT-116	Read MEP Database	CFM: M	12.14.7.6	Yes []
MGT-117	Read received Port Status TLV	CFM: O	item f) in 12.14.7.6.3, 20.19.3	Yes [] No []
MGT-118	Read received Interface Status TLV	CFM: O	item g) in 12.14.7.6.3, 20.19.4	Yes [] No []
MGT-119	Transmit MEP Fault Alarm	CFM: M	12.14.7.7	Yes []
MGT-120	List of active remote MEPs	CFM: O PBBTE: M	item ae) in 12.14.7.1.3	No [] N/A [] Yes []
MGT-121	Indication on the capability to report the Traffic field bit	PS-5: M	item af) in 12.14.7.1.3	Yes [] N/A []
MGT-122	Configuration to enable generation of a Fault Alarm based on a mismatch defect	PS-5: M	item ag) in 12.14.7.1.3	Yes [] N/A []
MGT-123	Indication on the presence of a traffic field mismatch defect	PS-5: M	item ah) in 12.14.7.1.3, 20.25.2	Yes [] N/A []
MGT-124	Indication on the presence of a local mismatch defect	PS-5: M	item ai) in 12.14.7.1.3, 20.25.5	Yes [] N/A []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-125	Indication on the presence of the mismatch defect since the MEP Mismatch Fault Notification Generator state machine was last in the MFNG_RESET state	PS-5: M	item aj) in 12.14.7.1.3	Yes [] N/A []
MGT-126	Read the current state of the MEP Mismatch Fault Notification Generator state machine	PS-5: M	item ak) in 12.14.7.1.3, 20.40	Yes [] N/A []
MGT-127	Configuration of the Reverse VID on LBM/LTM transmit on MAs associated with point-to-multipoint TESIs	PBBTE: M	item f) in 12.14.7.3.2, item e) in 12.14.7.4.2, 21.7.5.2	Yes [] N/A []
MGT-128	Read BEB configuration	MGT AND BEB: M	12.16.1.1	Yes [] N/A []
MGT-129	Set BEB configuration	MGT AND BEB: M	12.16.1.2	Yes [] N/A []
MGT-130	Create BEB component	MGT AND BEB: O	12.16.1.3	Yes [] N/A [] No []
MGT-131	Delete BEB component	MGT AND BEB: O	12.16.1.4	Yes [] N/A [] No []
MGT-132	Create BEB Bridge Port	MGT AND BEB: O	12.16.1.5	Yes [] N/A [] No []
MGT-133	Create BEB PIP	MGT AND BEB: O	12.16.1.6	Yes[] N/A [] No []
MGT-134	Delete BEB Bridge Port	MGT AND BEB: O	12.16.1.7	Yes [] N/A [] No []
MGT-135	Delete BEB PIP	MGT AND BEB: O	12.16.1.8	Yes [] N/A [] No []
MGT-136	Read BEB/PB/VLAN Bridge port configuration	MGT AND BEB: M	12.16.2.1	Yes [] N/A []
MGT-137	Read VIP configuration	MGT AND BEB-I: M	12.16.3.1	Yes [] N/A []
MGT-138	Set VIP configuration	MGT AND BEB-I: M	12.16.3.2	Yes [] N/A []
MGT-139	Read the value of the enableConnectionIdentifier parameter	MGT AND BEB-I: O PBBTE: M	item g) in 12.14.7.7.2, 6.10	No [] N/A [] Yes []
MGT-140	Configuration of the enableConnectionIdentifier parameter	MGT AND BEB-I: O PBBTE: M	item e) in 12.14.7.7.2, 6.10	No [] N/A [] Yes []
MGT-141	Read PIP configuration	MGT AND BEB-I: M	12.16.4.1	Yes [] N/A []
MGT-142	Set PIP configuration	MGT AND BEB-I: O	12.16.4.2	Yes [] N/A [] No []
MGT-143	Read VIP to PIP mapping	MGT AND BEB-I: M	12.16.4.3	Yes [] N/A []
MGT-144	Set VIP to PIP mapping	MGT AND BEB-I: O	12.16.4.4	Yes [] N/A [] No []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-145	Read PIP Priority Code Point Selection	MGT AND BEB-I: M	12.16.4.5	Yes [] N/A []
MGT-146	Set PIP Priority Code Point Selection	MGT AND BEB-I: M	12.16.4.6	Yes [] N/A []
MGT-147	Read PIP Priority Code Point Decoding Table	MGT AND BEB-I: M	12.16.4.7	Yes [] N/A []
MGT-148	Set PIP Priority Code Point Decoding Table	MGT AND BEB-I: O	12.16.4.8	Yes [] N/A [] No []
MGT-149	Read PIP Priority Code Point Encoding Table	MGT AND BEB-I: M	12.16.4.9	Yes [] N/A []
MGT-150	Set PIP Priority Code Point Encoding Table	MGT AND BEB-I: O	12.16.4.10	Yes[] N/A [] No []
MGT-151	Read PIP Use_DEI parameter	MGT AND BEB-I: M	12.16.4.11	Yes [] N/A []
MGT-152	Set PIP Use_DEI parameter	MGT AND BEB-I: O	12.16.4.12	YES[] N/A [] No []
MGT-153	Read PIP Require Drop Encoding parameter	MGT AND BEB-I: M	12.16.4.13	Yes [] N/A []
MGT-154	Set PIP Require Drop Encoding parameter	MGT AND BEB-I: O	12.16.4.14	Yes [] N/A [] No []
MGT-155	Read backbone service instance table entry	MGT AND BEB-B: M	12.16.5.1	Yes [] N/A []
MGT-156	Set backbone service instance table entry	MGT AND BEB-B: M	12.16.5.2	Yes [] N/A []
MGT-157	TESI assignment managed object	PBBTE: M	12.16.5.3	No [] N/A [] Yes []
MGT-158	Does the Bridge provide control of all the required DDCFM managed objects?	DDCFM:M	<12.17>	Yes []
	Questions MGT-159 through MGT-178 refer to operations related to DDCFM managed objects.			
MGT-159	Create Reflection Responder managed object	DDCFM:M	12.17.2.1	Yes []
MGT-160	Read Reflection Responder managed object	DDCFM:M	12.17.2.3	Yes []
MGT-161	Write Reflection Responder managed object's attributes	DDCFM:M	12.17.2.2	Yes []
MGT-162	Delete Reflection Responder managed object	DDCFM:M	12.17.2.4	Yes []
MGT-163	Activate a Reflection Responder managed object	DDCFM:M	12.17.2.5	Yes []
MGT-164	De-activate a Reflection Responder managed object	DDCFM:M	12.17.2.6	Yes []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-165	Create RFM Receiver managed object	DDCFM:M	12.17.3.1	Yes []
MGT-166	Delete RFM Receiver managed object	DDCFM:M	12.17.3.2	Yes []
MGT-167	Create Decapsulator Responder managed object	DDCFM:M	12.17.4.1	Yes []
MGT-168	Read Decapsulator Responder managed object	DDCFM:M	12.17.4.2	Yes []
MGT-169	Write Decapsulator Responder managed object's attributes	DDCFM:M	12.17.4.3	Yes []
MGT-170	Delete Decapsulator Responder managed object	DDCFM:M	12.17.4.4	Yes []
MGT-171	Activate a Decapsulator Responder managed object	DDCFM:M	12.17.4.5	Yes []
MGT-172	De-activate a Decapsulator Responder managed object	DDCFM:M	12.17.4.6	Yes []
MGT-173	Create SFM Originator managed object	DDCFM:M	12.17.5.1	Yes []
MGT-174	Read SFM Originator managed object	DDCFM:M	12.17.5.2	Yes []
MGT-175	Write SFM Originator managed object's attributes	DDCFM:M	12.17.5.4	Yes []
MGT-176	Delete SFM Originator managed object	DDCFM:M	12.17.5.3	Yes []
MGT-177	Activate SFM Originator managed object	DDCFM:M	12.17.5.5	Yes []
MGT-178	De-activate SFM Originator managed object	DDCFM:M	12.17.5.6	Yes []
MGT-179	Read TE protection group list	PBBTE: M	12.18.1.1	Yes [] N/A []
MGT-180	Create TE protection group managed object	PBBTE: M	12.18.1.2	Yes [] N/A [] N/A []
MGT-181	Delete TE protection group managed object	PBBTE: M	12.18.1.3	Yes [] N/A [] N/A []
MGT-182	Read TE protection group managed object	PBBTE: M	12.18.2.1	Yes [] N/A [] N/A []
MGT-183	Write TE protection group managed object	PBBTE: M	12.18.2.2	Yes [] N/A [] N/A []
MGT-184	TE protection group administrative commands managed object	PBBTE: M	12.18.2.3	Yes [] N/A [] N/A []
MGT-185	Does the implementation provide control of the TPMR management entity?	MGT AND TPMR: M	12.19.1	Yes [] N/A []
MGT-186	Does the implementation provide control of the individual MAC and PHY entities associated with each TPMR Port?	MGT AND TPMR: M		Yes [] N/A []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-187	Does the implementation provide control of the Forwarding Process of the MAC Relay Entity?	MGT AND TPMR: M	8.6, 12.19.3	Yes [] N/A []
MGT-188	Does the implementation provide control of the MAC status propagation entity?	MGT AND TPMR: M	23.3, 12.19.4	Yes [] N/A []
MGT-189	Read TPMR	TPMR:M	12.19.1.1.1	Yes [] N/A []
MGT-190	Set TPMR Name	TPMR:M	12.19.1.1.2	Yes [] N/A []
MGT-191	Read Port	TPMR:M	12.19.1.2.1	Yes [] N/A []
MGT-192	Set Port Name	TPMR:M	12.19.1.2.2	Yes [] N/A []
MGT-193	Read Forwarding Port Counters	TPMR:M	12.19.3.1.1	Yes [] N/A []
MGT-194	Read Port Default Priority	TPMR AND TPMR-12:M	12.6.2.1	Yes [] N/A []
MGT-195	Set Port Default Priority	TPMR AND TPMR-12:M	12.6.2.2	Yes [] N/A []
MGT-196	Read Port Priority Regeneration Table	TPMR AND TPMR-12:M	12.6.2.3	Yes [] N/A []
MGT-197	Set Port Priority Regeneration Table	TPMR AND TPMR-12:M	12.6.2.4	Yes [] N/A []
MGT-198	Read Port Traffic Class Table	TPMR AND TPMR-10:M	12.6.3.1	Yes [] N/A []
MGT-199	Set Port Traffic Class Table	TPMR AND TPMR-10:M	12.6.3.2	Yes [] N/A []
MGT-200	Read Port Priority Code Point Selection	TPMR AND TPMR-12:M	12.6.2.5	Yes [] N/A []
MGT-201	Set Port Priority Code Point Selection	TPMR AND TPMR-12:M	12.6.2.6	Yes [] N/A []
MGT-202	Read Priority Code Point Decoding Table	TPMR AND TPMR-12:O	12.6.2.7	Yes [] No [] N/A []
MGT-203	Read Use_DEI parameter	TPMR AND TPMR-12:M	12.6.2.12	Yes [] N/A []
MGT-204	Set Use_DEI parameter	TPMR AND TPMR-12:O	12.6.2.13	Yes [] No [] N/A []
MGT-205	Does the implementation support the management entities defined in 12.20?	FQTSS: O	5.4.1.5 item e), 12.20, 17.7.12, 34	Yes [] N/A []
MGT-206	Does the Bridge support all of the objects in the CN component managed object?	BRG1 AND CN: M	item c) in 5.4.3, 12.21.1	Yes []
MGT-207	Does the Bridge support all of the objects in the CN component priority managed object?	BRG1 AND CN: M	item c) in 5.4.3, 12.21.2	Yes []

A.14 Bridge management (*continued*)

Item	Feature	Status	References	Support
MGT-208	Does the Bridge support all of the objects in the CN Port priority managed object?	BRG1 AND CN: M	item c) in 5.4.3, 12.21.3	Yes []
MGT-209	Does the Bridge support all of the objects in the Congestion Point managed object?	BRG1 AND CN: M	item c) in 5.4.3, 12.21.4	Yes []
MGT-210	Does the implementation support the management entities defined in 12.22?	SRP: O	12.22	Yes [] N/A []

A.15 Remote management

Item	Feature	Status	References	Support
	If item RMGT is not supported, mark N/A and continue at A.16.			N/A[]
RMGT-1	What Management Protocol standard(s) or specification(s) are supported?	RMGT:M	5.4.1	
RMGT-2	What standard(s) or specifications for Managed Objects and Encodings are supported?	RMGT:M	5.4.1	

A.16 Expedited traffic classes

Item	Feature	Status	References	Support
TC-1	Does the implementation provide more than one transmission queue for (a) Bridge Port(s)?	TC: M	8.6.6	Yes [] N/A[]
TC-2	Is the recommended mapping of priority to traffic classes supported for each Port?	TC: O	8.6.6	Yes [] No []
TC-3	Can the traffic class tables be managed?	TC: O	8.6.6, Table 8-4	Yes [] No []
TC-4	Is the default algorithm for selecting frames for transmission supported?	M	8.6.8	Yes []
TC-5	Are additional algorithms for selecting frames for transmission supported?	O	8.6.8	Yes [] No []

A.17 Extended Filtering Services

Item	Feature	Status	References	Support
EFS-1	Can MAC Address Registration Entries be created, updated and removed from the Filtering Database by MMRP?	EFS:M	8.8, 8.8.4, Clause 10	Yes [] N/A[]
EFS-2	Can a Static Filtering Entry be created with an address specification that represents a Group Address, or All Group Addresses, or All Unregistered Group Addresses, and with a control element for each Port that specifies unconditional forwarding, or unconditional filtering, or the use of dynamic or default group filtering information?	EFS:M	8.8.1	Yes [] N/A[]
EFS-3	Can a Static Filtering Entry be created with an address specification that represents an Individual Address and with a control element for each Port that specifies unconditional forwarding, or unconditional filtering?	M	8.8.1	Yes []
EFS-4	Can a Static Filtering Entry be created with an address specification that represents an Individual Address and with a control element for each Port that specifies unconditional forwarding, or unconditional filtering, or the use of dynamic filtering information?	EFS:O	8.8.1	Yes [] No [] N/A[]

A.18 Multiple Spanning Tree Protocol

Item	Feature	Status	References	Support
	If item MSTP is not supported, mark N/A and continue at the start of A.19.			N/A []
MSTP-1	Support the CIST plus a stated maximum number of MSTIs, where that number is at least 2 and at most 64	MSTP:M	5.4.1, 8.8.8, 13.13	Yes []
MSTP-2	Support at least as many FIDs as MSTIs	MSTP:M	5.4, 5.4.1, 8.8.8	Yes []
MSTP-3	Associate each FID to a spanning tree	MSTP:M	5.4, 8.9.3	Yes []
MSTP-4	Transmit and receive MST Configuration Identifier information	MSTP:M	5.4, 8.9.2	Yes []
MSTP-5	Support a set of port state information per spanning tree per port	MSTP:M	5.4, 8.4, 13.36	Yes []
MSTP-6	Support an instance of spanning tree protocol per spanning tree per port	MSTP:M	5.4, 8.10, Clause 13	Yes []
MSTP-7	Use the Bridge Group Address as specified	MSTP:M	5.4, 8.13.2	Yes []
MSTP-8	Support default Bridge Forward Delay and Bridge Priority parameter values as specified	MSTP:M	5.4, 13.24	Yes []
MSTP-9	Provision of identifiers for Bridge and Ports	MSTP:M	13.2	Yes []
MSTP-10	Not exceed the values in {D}17.32 for max BPDU transmission delay	MSTP:M	{D}Table 17-5	Yes []

A.18 Multiple Spanning Tree Protocol (continued)

Item	Feature	Status	References	Support
MSTP-11	Inclusion of active Ports in computation of the active topology for a given spanning tree	MSTP:M	13.8	Yes []
MSTP-12	Processing of BPDUs received on Ports included in the computation of the active topology for a given spanning tree	MSTP:M	13.8	Yes []
MSTP-13	Discarding received frames in the Discarding state for a given spanning tree	MSTP:M	13.8	Yes []
MSTP-14	Incorporating station location information to the Filtering Database in the Learning and Forwarding states for a given spanning tree	MSTP:M	13.8	Yes []
MSTP-15	Not incorporating station location information to the Filtering Database in the Discarding state for a given spanning tree	MSTP:M	13.8	Yes []
MSTP-16	Transfer learned MAC addresses from a retiring Root Port to a new Root Port for a given spanning tree	MSTP:O	13.15	Yes [] No []
MSTP-17	Instances of state machines per Bridge, per Port and per spanning tree instance, as specified	MSTP:M	13.22	Yes []
MSTP-18	Port Timers state machine support	MSTP:M	13.23, 13.28	Yes []
MSTP-19	Port Receive state machine support	MSTP:M	13.23, 13.29	Yes []
MSTP-20	Port Protocol Migration state machine support	MSTP:M	13.23, 13.30	Yes []
MSTP-21	Port Transmit state machine support	MSTP:M	13.23, 13.32	Yes []
MSTP-22	Port Information state machine support	MSTP:M	13.23, 13.33	Yes []
MSTP-23	Port Role Selection state machine support	MSTP:M	13.23, 13.34	Yes []
MSTP-24	Port Role Transitions state machine support	MSTP:M	13.23, 13.35	Yes []
MSTP-25	Port State Transition state machine support	MSTP:M	13.23, 13.36	Yes []
MSTP-26	Topology Change state machine support	MSTP:M	13.23, 13.37	Yes []
MSTP-27	Are the values of Migrate Time, Bridge Hello Time, Bridge Max Age, Bridge Forward Delay, Transmit Hold Count and Max Hops within the permitted range?	MSTP:M	13.23, Table 13-5	Yes []
MSTP-28	Enforcement of parameter relationships	MSTP:M	13.23, Table 13-5	Yes []
MSTP-29	Range and granularity of priority values	MSTP:M	13.16	Yes []
MSTP-30	Range and granularity of path cost values	MSTP:M	13.16	Yes []
MSTP-31	Are all Provider Network Ports capable of supporting the L2GP spanning tree protocol?	PB: O	13.38	Yes [] N/A [] No []
MSTP-32	Are Provider Instance Ports capable of supporting the encapsulation/decapsulation of BPDUs?	BEB-I: O	5.7.1	Yes [] N/A [] No []

A.18 Multiple Spanning Tree Protocol (*continued*)

Item	Feature	Status	References	Support
MSTP-33	Can each Port be configured to support detection of fragile bridges by setting the autoIsolatePort parameter?	MSTP:O	13.21, 13.25.6	Yes [] No []
MSTP-34	Can a port be configured as an L2 Gateway Port by setting the isL2gp parameter?	MSTP:O	13.18, 13.25.19	Yes [] No []
MSTP-35	Can the pseudoRootId parameter be set for an L2 Gateway Port for the CIST and for each MSTI?	MSTP-34:M	13.18, 13.25.37	Yes [] N/A []
MSTP-36	Is one instance of the L2 Gateway Port state machine implemented for each L2GP port?	MSTP-34:M	13.38	Yes [] N/A []

A.19 VLAN support

Item	Feature	Status	References	Support
VLAN-1	Does the implementation support, on each Port, one or more of the permissible combinations of values for the Acceptable Frame Types parameter?	M	5.4, 6.9	Yes []
VLAN-2	Does the implementation support, on each Provider Bridge Port, the <i>Admit All Frames</i> value for the Acceptable Frame Types Parameter?	PB:M	5.6, 6.9	Yes [] N/A []
VLAN-3	State which Ports support the following values for the Acceptable Frame Types parameter: — <i>Admit Only VLAN-tagged frames</i> ; — <i>Admit Only Untagged and Priority Tagged frames</i> ; — <i>Admit All frames</i> .	M	5.4, 6.9	Ports: _____ Ports: _____ Ports: _____
VLAN-4	On Ports that support both values for the Acceptable Frame Types parameter, is the parameter configurable via management?	M	5.4, 6.9, 12.10	Yes [] N/A []
VLAN-5	Does the implementation support the ability for the Filtering Database to contain static and dynamic configuration information for at least one VID, by means of Static and Dynamic VLAN Registration Entries?	M	5.4, 8.8	Yes []
VLAN-6	Does the implementation support at least one FID?	M	5.4, 8.8.3, 8.8.8, 8.8.9	Yes []
VLAN-7	Can the implementation allocate at least one VID to each FID supported?	M	5.4, 8.8.3, 8.8.8, 8.8.9	Yes []
VLAN-8	Does the implementation take account of the allocation of VIDs to FIDs when making forwarding decisions relative to group MAC Addresses?	O	8.8.9	Yes [] No []

A.19 VLAN support (*continued*)

Item	Feature	Status	References	Support	
VLAN-9	On Ports that support untagged and priority-tagged frames, does the implementation support: — A PVID value? — The ability to configure one VID whose untagged set includes that Port? — Configuration of the PVID value via management operations? — Configuration of Static Filtering Entries via management operations? — The ability to configure more than one VID whose untagged set includes that Port?	M M M M O	5.4, 6.9, 8.8.2, 12.10	Yes [] N/A [] Yes [] No [] N/A []	
VLAN-10	Does the implementation support the ability to enable and disable Ingress Filtering?	CB:O PB:M BEB:M	5.4.1, 5.6, 8.6.3	Yes[]	No [] N/A []
VLAN-11	Can the PVID or the VID in any member of the VID Set for any Port be assigned the value of the null VLAN ID?	X	6.9, Table 9-2	No []	
VLAN-12	Are frames discarded (or not discarded) in accordance with the settings of the Acceptable Frame Types parameters?	M	6.9	Yes []	
VLAN-13	Are all frames received classified as belonging to exactly one VID, as defined in the ingress rules?	M	6.9	Yes []	
VLAN-14	Is Ingress Filtering performed in accordance with the value of the Enable Ingress Filtering parameter?	M	8.6.2	Yes []	
VLAN-15	Are all frames that are not discarded as a result of the application of the ingress rules submitted to the Forwarding Process and to the Learning Process?	M	8.6.2	Yes []	
VLAN-16	Does the implementation support Port-and-Protocol-based classification of frames on any or all Ports?	O	6.12	Yes []	No []
VLAN-16.1	State which Ports support Port-and-Protocol-based classification rules.	VLAN-16:M	6.12	Ports:	_____
VLAN-16.2	For each Port that supports Port-and-Protocol-based classification rules, is a VID Set supported?	VLAN-16:M	6.12	Port: Yes []	N/A []
VLAN-16.3	For each Port that supports Port-and-Protocol-based classification rules, state how many entries are supported in the VID Set.	VLAN-16:M	6.12	Port: _____	Entries
VLAN-16.4	For each Port that supports Port-and-Protocol-based classification rules, is the VID Set configurable via management?	VLAN-16:M	12.10.1.2	Port: Yes []	N/A []
VLAN-17	Does the implementation support a Protocol Group Database?	VLAN-16:M	6.12.3	Yes []	N/A []

A.19 VLAN support (*continued*)

Item	Feature	Status	References	Support
VLAN-17.1	State how many entries are supported in the Protocol Group Database.	VLAN-17:M	6.12.3	Entries _____
VLAN-17.2	Is the Protocol Group Database configurable via management?	VLAN-17:O	12.10.2.1	Yes [] No []
VLAN-17.3	Does the Protocol Group Database support entries of format Ethernet?	VLAN-17:O	6.12.3	Yes [] No []
VLAN-17.4	Does the Protocol Group Database support entries of format RFC_1042?	VLAN-17:O	6.12.3	Yes [] No []
VLAN-17.5	Does the Protocol Group Database support entries of format SNAP_8021H?	VLAN-17:O	6.12.3	Yes [] No []
VLAN-17.6	Does the Protocol Group Database support entries of format SNAP_Other?	VLAN-17:O	6.12.3	Yes [] No []
VLAN-17.7	Does the Protocol Group Database support entries of format LLC_Other?	VLAN-17:O	6.12.3	Yes [] No []
VLAN-17.8	Does the Protocol Group Database support entries of at least one of the following formats: Ethernet, RFC_1042, SNAP_8021H, SNAP_Other, LLC_Other?	VLAN-17: M	6.12.3	Yes []
VLAN-18	Are frames discarded if the transmission Port is not present in the member set for the frame's VID?	M	6.9.2, 8.8.10	Yes []
VLAN-19	Are frames transmitted as VLAN-tagged frames or as untagged frames in accordance with the value of the untagged set for the frame's VID?	M	8.8.2	Yes []
VLAN-20	Does the implementation support Static VLAN Registration Entries as defined in 8.8.2?	M	8.8.2	Yes []
VLAN-21	Does the implementation support the creation of a separate Static VLAN Registration Entry with a distinct Port Map for each VID from which frames are received by the Forwarding Process?	O	8.8.2	Yes [] No []
VLAN-22	Does the implementation support Dynamic VLAN Registration Entries as defined in 8.8.5?	M	8.8.5	Yes []
VLAN-23	Does the implementation support the creation of a separate Dynamic VLAN Registration Entry with a distinct Port Map for each VID from which frames are received by the Forwarding Process?	O	8.8.5	Yes [] No []
VLAN-24	Does the implementation allocate VIDs to FIDs in accordance with the specification in 8.8.8?	M	8.8.8, 8.8.8.2	Yes []
VLAN-25	Does the implementation correctly detect Learning Constraint violations?	M	8.8.8.3	Yes []

A.19 VLAN support (continued)

Item	Feature	Status	References	Support
VLAN-26	Is determination of the member set and the untagged set for a given VID achieved as defined in 8.8.10?	M	8.8.10	Yes []
VLAN-27	Do VLAN-tagged frames transmitted by the Bridge conform to the format defined in Clause 9 for the MAC type on which they are transmitted?	M	Clause 9	Yes []
VLAN-28	Are all BPDUs transmitted untagged?	M	8.13.9	Yes []
VLAN-29	Is encoding of the drop_eligible parameter in the Priority Code Point field of the VLAN tag supported?	CB:O PB:M BEB:M	5.5.1, 5.6, 6.9.3, 9.6	Yes [] No [] N/A []
VLAN-30	Is Service Access Priority Selection supported?	CB:O PB:X BEB:X	5.5.1, 5.6, 6.13	Yes [] No [] N/A []
VLAN-31	Is the VID translation table supported?	CB:X PB:O BEB:O	5.5, 5.6.1, 6.9	Yes [] No [] N/A []
VLAN-32	Is the VIP-ISID parameter supported?	BEB-I: M	6.10	Yes [] N/A []
VLAN-33	Is the default Backbone Destination supported?	BEB-I: M BEBTE: M	6.10	Yes [] N/A []
VLAN-34	Is the adminPointToPointMAC supported?	BEB-I: O	6.10	Yes [] No [] N/A []
VLAN-35	Is the Backbone Service Instance field in the backbone service instance table supported?	BEB-B: M	6.11	Yes [] N/A []
VLAN-36	Is the Backbone VLAN identifier field in the backbone service instance table supported?	BEB-B: O BEBTE: M	6.11	Yes [] No [] N/A []
VLAN-37	Is the Local Service Instance field in the backbone service instance table supported?	BEB-B: O	6.11	Yes [] No [] N/A []
VLAN-38	Is the Default Backbone Destination field in the backbone service instance table supported?	BEB-B: O BEBTE: M	6.11	Yes [] No [] N/A []
VLAN-39	Is many-to-one S-VID to I-SID mapping supported?	BEB-I: O	5.7.1	Yes [] No [] N/A []

A.20 MMRP

Item	Feature	Status	References	Support
	If MMRP is not supported, mark N/A and continue at A.21			N/A []
MMRP1	Does the implementation support the exchange of MMRPDUs, using the generic MRPDU format defined in 10.8 to exchange MMRP-specific information, as defined in 10.12?	M	5.4.1.3, 10.8, 10.12	Yes []
MMRP2	Is the MMRP Application supported as defined in 10.12?	M	5.4.1.3, 10.12	Yes []
MMRP3	Does the implementation propagate registration information in accordance with the operation of MAP for the Base Spanning Tree Context, as specified in 10.3.1?	M	5.4.1.3, 10.3.1	Yes []
MMRP4	Does the implementation forward, filter, or discard MAC frames carrying any MRP Application address as the destination MAC address in accordance with the requirements of 8.13.6?	M	5.4.1.3, 8.13.6	Yes []
MMRP5	Is the MAP Context Identifier used to identify a VLAN Context equal to the VID used to identify the corresponding VLAN?	M	10.12.1.1	Yes []
MMRP6	Is the set of Ports of a Bridge defined to be part of the active topology for a given VLAN Context as specified in 10.12.1.1?	M	10.12.1.1	Yes []
MMRP7	Is the group MAC address used as the destination address for MRPDUs destined for MMRP Participants the group MAC address identified in Table 10-1 as “Customer and Provider Bridge MMRP address”?	M	10.12.1.3	Yes []
MMRP8	Is the EtherType used for MRPDUs destined for MMRP Participants the MMRP EtherType identified in Table 10-2?	M	10.12.1.4, Table 10-2	Yes []
MMRP9	Does the ProtocolVersion used for the implementation of MMRP take the hexadecimal value 0x00?	M	10.12.1.5	Yes []
MMRP10	Are the Attribute Type values used in the implementation as specified in 10.12.1.6?	M	10.12.1.6	Yes []
MMRP11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 10.12.1.7?	M	10.12.1.7	Yes []
MMRP12	Is management of the Restricted_MAC_Address_Registration control parameter supported?	O	10.12.2	Yes [] No []
MMRP13	If management of the Restricted_MAC_Address_Registration control parameter is not supported, is the value of this parameter FALSE for all Ports?	¬MMRP12:M	10.12.2	Yes [] N/A []

A.20 MMRP (*continued*)

Item	Feature	Status	References	Support
MMRP14	Does the implementations maintain state information for all attribute values that support the Group service requirement registration?	M	10.10.2	Yes []
MMRP15	Is the implementation capable of supporting any attribute value in the range of possible values that can be registered using Group membership and individual MAC address registration?	M	10.12.4	Yes []
MMRP16	State the number of Group membership and individual MAC address state values that can be supported on each Port.	M	10.12.4	Number _____

A.21 MVRP

Item	Feature	Status	References	Support
	If MVRP is not supported, mark N/A and continue at A.22			N/A []
MVRP1	Does the implementation support the exchange of MMRPDUs, using the generic MRPDU format defined in 11.2 to exchange MMRP-specific information, as defined in 10.12?	M	5.4.2, 10.8, 11.2	Yes []
MVRP2	Is the MMRP Application supported as defined in 11.2?	M	5.4.2, 11.2	Yes []
MVRP3	Does the implementation propagate registration information in an MST Bridge, in accordance with the operation of MAP for multiple Spanning Tree Contexts, as specified in 11.2.3.1.2?	MSTP:M	5.4.2, 11.2.3.1.2	Yes [] N/A []
MVRP4	Does the implementation propagate registration information in an SST Bridge, in accordance with the operation of MAP for the Base Spanning Tree Context, as specified in 10.3.1?	¬MSTP:M	5.4.2, 10.3.1	Yes [] N/A []
MVRP5	Does the implementation forward, filter or discard MAC frames carrying any MRP Application address as the destination MAC address in accordance with the requirements of 8.13.6?	M	5.4.2, 8.13.6	Yes []
MVRP6	If a VID translation table is in use, does the implementation translate VIDs in MVRPDUs as specified in 11.2.4?	VTR:M	11.2.4	Yes []
MVRP7	Is the group MAC address used as the destination address for MRPDUs destined for MVRP Participants as defined in 11.2.3.1.3?	M	11.2.3.1.3	Yes []
MVRP8	Is the EtherType used for MRPDUs destined for MVRP Participants the MVRP EtherType identified in Table 10-2?	M	11.2.3.1.4, Table 10-2	Yes []
MVRP9	Does the ProtocolVersion used for the implementation of MVRP take the hexadecimal value 0x00?	M	11.2.3.1.5	Yes []
MVRP10	Are the Attribute Type values used in the implementation as specified in 11.2.3.1.6?	M	11.2.3.1.6	Yes []
MVRP11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 11.2.3.1.7?	M	11.2.3.1.7	Yes []
MVRP12	Is the implementation of MVRP capable of supporting all attribute values in the range of possible values that can be registered using MVRP?	M	11.2.6	Yes []
MVRP13	Is the implementation capable of maintaining current state information for all attributes in the range of possible values?	M	11.2.6	Yes []

A.22 MRP

Item	Feature	Status	References	Support
MRP1	Does the implementation of MRP meet all of the requirements for interoperability stated in 10.5 that apply to Bridge operation?	M	10.5	Yes []
MRP2	Does the implementation support the operation of the complete Applicant state machine?	MRP1:M	10.7, 10.7.7	Yes [] N/A []
MRP3	Does the implementation support the operation of the point-to-point subset of the Applicant state machine?	MRP2:M	10.7, 10.7.7	Yes [] N/A []
MRP4	Does the implementation support the operation of the Registrar state machine?	M	10.7, 10.7.8	Yes []
MRP5	Does the implementation support the operation of the LeaveAll state machine?	M	10.7, 10.7.9	Yes []
MRP6	Does the implementation support the operation of the PeriodicTransmission state machine?	M	10.7, 10.7.10	Yes []

A.23 Connectivity Fault Management

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 5.4.1.4	Yes []
CFM-2	Does the Bridge support the creation of a Maintenance Association (MA) on each VID supported by the Bridge for each MD Level?	BRG AND CFM: M	item b) in 5.4.1.4	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 5.4.1.4	Yes []
CFM-4	Does the Bridge support the creation of eight Up MEPs on each VID on each Port, each MEP at a different MD Level?	BRG AND CFM: M	item d) in 5.4.1.4	Yes []
CFM-5	Does the Bridge support the creation of eight Down MEPs on each VID on each Port, each MEP at a different MD Level?	BRG AND CFM: M	item e) in 5.4.1.4	Yes []
CFM-6	Does the Bridge support the creation of eight Down MEPs associated with no VID on each Port, each MEP at a different MD Level?	BRG AND CFM: M	item f) in 5.4.1.4	Yes []
CFM-7	Does the Bridge support the maintenance of a MEP CCM Database?	BRG AND CFM: M	item g) in 5.4.1.4	Yes []
CFM-8	Does the Bridge conform to the state machines and procedures in Clause 20?	BRG AND CFM: M	item i) in 5.4.1.4, 20	Yes []

A.23 Connectivity Fault Management (*continued*)

Item	Feature	Status	References	Support
CFM-9	Does the Bridge transmit and accept frames in the formats specified in Clause 21?	BRG AND CFM: M	item j) in 5.4.1.4	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 5.4.1.4, 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 5.4.1.4, 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 5.4.1.4, 19.3, 19.2.8	Yes [] No []
CFM-13	Does the Bridge support the creation of MAs that are associated with more than one VID?	BRG AND CFM: O	item o) in 5.4.1.4	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 5.4.35.17	Yes []
CFM-15	Does the Station support the creation of a Maintenance Association (MA) on each VID supported by the Bridge for each MD Level?	¬BRG AND CFM: M	item b) in 5.4.35.17	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 VID on each Port, each MEP at a different MD Level?	¬BRG AND CFM: M	item c) in 5.4.35.17	Yes []
CFM-19	Does the Station support the creation of eight Down MEPs associated with no VID on each Port, each MEP at a different MD Level?	¬BRG AND CFM: M	item d) in 5.4.35.17	Yes []
CFM-20	Does the Station support the maintenance of a MEP CCM Database?	¬BRG AND CFM: M	item e) in 5.4.35.17	Yes []
CFM-21	Does the Station conform to the state machines and procedures in Clause 20?	¬BRG AND CFM: M	item g) in 5.4.35.17, 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 5.4.35.17	Yes []
CFM-23	Does the Station support the creation of MAs that are associated with more than one VID?	¬BRG AND CFM: O	item j) in 5.4.35.17	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, 20.51.4.1	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 []

A.23 Connectivity Fault Management (*continued*)

Item	Feature	Status	References	Support
CFM-26	Are the contents of an LBM, except for the source_address and OpCode, ignored and not interpreted by a non-PBB-TE receiver?	~CFM-91 AND CFM: M	20.2.2	Yes [] N/A []
CFM-27	Is the LTFwhile timer implemented?	CFM:M	20.5, 20.5.1	Yes []
CFM-28	Are the per-MEP timers CCIwhile, errorCCMwhile, 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, 20.5.10	Yes []
CFM-30	Can a MEP set rMEPCCMdefect within $(3.25 * \text{CCMtime}(\text{CCMinterval}))$ seconds of the receipt of a CCM?	X	20.5.10	No []
CFM-31	Can a MEP take longer to set rMEPCCMdefect than $(3.5 * \text{CCMtime}(\text{CCMinterval}))$ seconds of the receipt of the last CCM?	CFM: X	20.5.10	No [] N/A []
CFM-32	Does the system transmit a nonzero 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	20.51.2	Yes []
CFM-34	Does the Bridge transmit all reserved bits and fields in CFM PDUs as 0?	CFM: M	20.51.2	Yes []
CFM-35	Does the Bridge transmit additional fixed header fields not defined in this standard in CFM PDUs?	X	20.51.2	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 (02/2008)?	X	20.51.2	No []
CFM-37	Does the Bridge transmit additional fields in any CFM PDU in any TLV defined by this standard?	X	20.51.2	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	20.51.3	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, 20.28.1, 20.33.1, 20.51.4.2	Yes []

A.23 Connectivity Fault Management (*continued*)

Item	Feature	Status	References	Support
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, 20.28.1, 20.33.1, 20.51.4.1, 20.51.4.3	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	20.51.2, 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 []
CFM-46	Do the Organization-Specific TLV(s) transmitted by the Bridge conform to the validation and versioning rules of 20.51?	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 a 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 []

A.23 Connectivity Fault Management (*continued*)

Item	Feature	Status	References	Support
CFM-55	Is the field Defined by ITU-T Y.1731 (02/2008) 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 []
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	20.33.1, 20.44.1, 20.47.1	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, 20.28.1	Yes [] No []
CFM-70	Does the Bridge check the validity of every received LBR?	CFM: O	20.2.2, 20.33.1	Yes [] No []

A.23 Connectivity Fault Management (*continued*)

Item	Feature	Status	References	Support
CFM-71	Does the Bridge compare the received LBR bit-by-bit against the original LBM?	CFM: O	20.2.3, 20.33.1	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	20.51.5	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 []
CFM-81	Can CFM monitor backbone service instances using Backbone Service Instance Multiplex Entities placed back-to-back?	BEB: O	5.7.1, 5.8.1	Yes [] N/A [] No []
CFM-82	Does the Bridge support the creation of a Maintenance Association (MA) on each TESI supported by the Bridge for each MD Level?	BEBTE: M	5.8.2, 26.9	Yes [] N/A []
CFM-83	Does the Bridge support the creation of an Up MEP on each TESI on each CBP?	BEBTE: M	5.8.2, 6.19, 26.9.3	Yes [] N/A []
CFM-84	Does the Bridge support the creation of eight Up MEPs on each TESI on each CBP, each MEP at a different MD Level?	BEBTE: O	5.8.2	Yes [] No [] N/A []
CFM-85	Does the Bridge support the creation of MIPs on TESIs?	PBBTE: O	5.6.2, 5.8.2	Yes [] No [] N/A []
CFM-86	Is the PBB-TE MIP TLV included in every LBM that is targeting a PBB-TE MIP?	BEBTE: M	20.2, 21.7.5	Yes [] N/A []
CFM-87	Is the PBB-TE MIP TLV included in every LTM that is associated with a PBB-TE MA?	BEBTE: M	20.3, 21.7.5	Yes [] N/A []
CFM-88	Is the PBB-TE MIP TLV included in LBMs that are targeting a PBB-TE MEP?	BEBTE: X	20.2, 21.7.5	No [] N/A []

A.23 Connectivity Fault Management (*continued*)

Item	Feature	Status	References	Support
CFM-89	Is the PBB-TE MIP TLV the first TLV in an LBM targeting a PBB-TE MIP?	CFM-86: M	20.2, 21.7.5	Yes [] N/A []
CFM-90	Is the Reverse MAC field included in the PBB-TE MIP TLVs required by LBMs/LTMs that are associated with Point-to-Multipoint PBB-TE MAs?	CFM-86 OR CFM-87: M	20.2, 20.3, 21.7.5	Yes [] N/A []
CFM-91	Does the PBB-TE MEP check the MAID in received CCMs?	BEBTE: O	20.1.3, 20.17.1	Yes [] N/A []
CFM-92	Does the PBB-TE MEP set the Traffic field bit in the transmitted CCMs and process it in the received CCMs	PS-5: M	20.11.1, 20.17.1, 21.6.1.4	Yes [] N/A []
CFM-93	Does the Bridge keep track of the Traffic field bit in received CCMs?	PS-5: M	20.16.3, 20.17.1, 21.6.1.4	Yes [] N/A []
CFM-94	Are the per-PBB MEP timers mmCCMwhile, mmLocwhile and mmFNGwhile implemented?	PS-5: M	20.5.7	Yes [] N/A []
CFM-95	Are the per-PBB MEP mismatch defect related, MEP Traffic Field Mismatch state machine, MEP Local Mismatch state machine and MEP Mismatch Fault Notification Generator state machine implemented?	PS-5: M	20.24, 20.40	Yes [] N/A []

A.24 Management Information Base (MIB)

Item	Feature	Status	References	Support
	If item MIB is not supported, mark N/A			N/A []
MIB-1	Is the IEEE8021-TC-MIB module supported?	MIB:M	17.7.1	Yes [] No []
MIB-2	Is the IEEE8021-BRIDGE-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB:M	17.7.2	Yes [] No []
MIB-3	Is the IEEE8021-SPANNING-TREE-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB:O	17.7.3	Yes [] No []
MIB-5	Is the IEEE8021-Q-BRIDGE-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB:M	17.7.4	Yes [] No []
MIB-6	Is the IEEE8021-PB-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND PB:O	17.7.5	Yes [] No []
MIB-7	Is the IEEE8021-MST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND MSTP:O	17.7.6	Yes [] No []
MIB-8	Is the IEEE8021-CFM-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND CFM:O	17.7.7	Yes [] No []

A.24 Management Information Base (MIB) (continued)

Item	Feature	Status	References	Support
MIB-9	What method is used by the Bridge to provide control of all of the required CFM managed objects?	CFM: O.3	item n in , 17.3.7, 17.7.7	Yes [] No []
MIB-10	The CFM Management Information Base (MIB) module defined in 17.7.7?	CFM: O.3	17.3.7	Yes [] No []
MIB-11	Some other method than the MIB module defined in 17.7.7? If by some other method, what method is used?	MIB-10: M	17.3.7	_____
MIB-12	Does the Station provide control of all of the required CFM managed objects?	~BRG AND CFM: M	item f in 5.4.3, 12.14	Yes []
MIB-13	What method is used by the Station to provide control of all of the required CFM managed objects?	~BRG AND CFM: O.4	item i in 5.4.3, 17.3.7, 17.7.7	Yes [] No []
MIB-14	The CFM Management Information Base (MIB) module defined in 17.7.7?	~BRG AND CFM: O.4	17.3.7	Yes [] No []
MIB-15	Some other method than the MIB module defined in 17.7.7? If by some other method, what method is used?	MIB-14: M	17.3.7	_____
MIB-16	If an entire C-tagged service interface given a single row in the IETF RFC 2863 IF-MIB?	PEB AND CFM: O	17.3.7	Yes [] No []
MIB-17	Is every Bridge Port assigned its own conceptual row in the IETF RFC 2863 IF-MIB with its own unique ifIndex?	MIB-13: M	17.3.7	Yes []
MIB-18	Does the Bridge support IEEE 802.1AX Link Aggregation?	CFM: O	17.3.7, IEEE Std 802.1AX- 2008	Yes [] No []
MIB-19	Does every IEEE 802.3 MAC, when aggregated via IEEE 802.1AX Link Aggregation, have its own unique ifIndex, separate from the ifIndex of the Bridge Port as a whole?	MIB-18: M	17.3.7	Yes [] No []
MIB-20	Is the IEEE8021-PBB-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND BEB:O	17.7.8	Yes [] No []
MIB-21	Is the IEEE8021-DDCFM-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND DDCFM:O	17.7.9	Yes[]
MIB-22	Is the IEEE8021-PBBTE-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND PBBTE: O	17.7.10	Yes [] No [] N/A []
MIB-23	Is the IEEE8021-TPMR-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND TPMR:O	17.7.11	Yes [] No [] N/A []
MIB-24	Is the IEEE8021-FQTSS-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND FQTSS: O	5.4.1.5 item e), 12.20, 17.7.12, 34	Yes [] N/A []

A.24 Management Information Base (MIB) (continued)

Item	Feature	Status	References	Support
MIB-25	Does the system implement the IEEE8021-CN-MIB?	MIB AND CN: O	item j) in 5.4.3, item n) in 5.19	Yes [] No []
MIB-26	Does the system implement the System Group and Interfaces Group of the SNMPv2-MIB?	MIB-25: M	17.3.7.1	Yes []
MIB-27	Does the system implement the clarifications of the Interfaces group supplied in the descriptions of ieee8021CnEplIfIndex, ieee8021CnPortPriIfIndex, ieee8021CnCpIfIndex, and ieee8021CnCpidToIfIfIndex?	MIB-25: M	17.3.7.1, 17.7.13	Yes []
MIB-28	Does the system assign the same conceptual row in the IF-MIB to both an aggregated port and to one of the ports being aggregated?	MIB-25: M	17.3.7.1	No []
MIB-29	Does the system assign a conceptual row in the IF-MIB to the individual aggregated?	MIB-25: O	17.3.7.1	Yes [] No []
MIB-30	Does the system allow more than seven rows to be created in the ieee8021CnComptPriTable?	MIB-25: M	17.7.13	No []
MIB-31	Is the minimum value for ieee8021CnRpgMaxRate supported by the system at least equal to the required value?	MIB-25: M	17.7.13, 32.11.5	Yes []
MIB-32	Is the minimum value for ieee8021CnRpgMinRate supported by the system at least equal to the required value?	MIB-25: M	17.7.13, 32.11.10	Yes []
MIB-33	Is the IEEE8021-SRP-MIB module fully supported (per its MODULE-COMPLIANCE)?	SRP: O	17.2, 35	Yes [] N/A []

A.25 Protection Switching

Item	Feature	Status	References	Support
	If item PS is not supported, mark N/A and continue at the subsequent subclause.			N/A []
PS-1	Is 1:1 protection switching supported?	PS: M	5.8.2, 26.10.3	Yes [] N/A []
PS-2	Are the operator commands Forced Switch, Lockout of Protection, MStoWorking and MStoProtection implemented?	PS-1: M	26.10.3.3	Yes [] No [] N/A []
PS-3	Is the hold-off timer implemented?	PS-1: O	26.10.3.2.2	Yes [] No [] N/A []
PS-4	Is protection switching with load sharing supported?	PS: O	5.8.2, 12.14.1.2	Yes [] No [] N/A []
PS-5	Is the detection of mismatch defects supported?	PS: O	26.11	Yes [] No [] N/A []

A.26 Data-driven and data-dependent connectivity fault management

Item	Feature	Status	References	Support
DDCFM-1	Does the Bridge support the Reflection Responder function?	DDCFM:M	29.2.2	Yes []
DDCFM-2	Does the Bridge support the RFM Receiver function?	DDCFM:M	29.2.4	Yes []
DDCFM-3	Does the Bridge support the Decapsulator Responder function?	DDCFM:M	29.2.6	Yes []
DDCFM-4	Does the Bridge support the SFM Originator function?	DDCFM:M	29.2.7	Yes []

A.27 TPMR

Item	Feature	Status	References	Support
	If TPMR is not supported, mark N/A and continue at A.28			N/A []
TPMR-1	Does the TPMR comprise a single TPMR component?	TPMR:M	5.13	Yes []
TPMR-2	Is each TPMR Port capable of attaching directly to an IEEE 802 LAN?	TPMR:M	5.14	Yes []
TPMR-3	Does the implementation support exactly two externally-accessible Bridge Ports?	TPMR:M	5.13	Yes []
TPMR-4	Does the implementation support the operation of the Bridge Port Transmit and Receive process, using the TPMR Port connectivity rules?	TPMR:M	5.13, 8.5, 8.5.2	Yes []
TPMR-5	Does the Forwarding Process support frame filtering, queuing frames, queue management and transmission selection?	TPMR:M	5.13, 8.6.3, 8.6.6, 8.6.7, 8.6.8	Yes []
TPMR-6	Does the implementation support at least one traffic class on each externally accessible Port?	TPMR:M	5.13	Yes []
TPMR-7	Does the implementation support the management functionality specified in 12.19?	TPMR:M	12.19	Yes []
TPMR-8	Does the implementation support remote management via one of the externally accessible Ports?	TPMR:M	5.13, 8.13.7	Yes []
TPMR-9	Does the implementation support static filtering entries in accordance with the requirements stated in 5.13:f?	TPMR:M	5.13, 8.13.7, Table 8-3	Yes []
TPMR-10	Does the implementation support multiple traffic classes on each externally accessible Port?	TPMR:O	5.13	Yes [] No []
TPMR-11	Does the implementation support remote management via one or both of the externally accessible Ports, using IETF RFC 4789, and the TPMR MIB module?	TPMR:O	5.13, IETF RFC 4789, 8.13.7, 17.7.11	Yes [] No []
TPMR-12	Does the implementation support signaled priority?	TPMR: O	6.20	Yes [] No [] N/A []

A.28 MSP

Item	Feature	Status	References	Support
	If MSP is not supported, mark N/A			N/A []
MSP-1	Does the implementation support the operation of the Status Transmission state machine	MSP:M	23.8	Yes []
MSP-2	Does the implementation support the operation of the Status Notification state machine	MSP:M	23.9	Yes []
MSP-3	Does the Receive Process receive and validate MSPDUs as specified?	MSP:M	23.10, 23.16	Yes []
MSP-4	Does the Transmit Process identify, address, structure, encode and transmit MSPDUs as specified?	MSP:M	23.11, 23.13, 23.14, 23.15	Yes []
MSP-5	Does the implementation allow performance parameters to be read by management?	MSP:O	23.12	Yes [] No []
MSP-6	Does the implementation allow performance parameters to be modified by management?	MSP:O	23.12	Yes [] No []
MSP-7	Does the implementation maintain the specified counts for at least one Port of the TPMR?	MSP:O	23.12	Yes [] No []
MSP-8	Does the implementation maintain the specified counts for both ports of the TPMR?	MSP:O	23.12	Yes [] No []

A.29 Forwarding and queuing for time-sensitive streams

Item	Feature	Status	References	Support
	If forwarding and queuing for time-sensitive streams (FQTSS in Table A.6) is not supported, mark N/A and ignore the remainder of this table.		5.4.1.5, 34	N/A[]
FQTSS1	Support a minimum of two traffic classes, of which one supports the strict priority algorithm and the other is an SR class.	FQTSS:M	5.4.1.5, 8.6.8.1, 34	Yes [] N/A []
FQTSS2	Support the operation of the credit-based shaper algorithm on all Ports as the transmission selection algorithm used for the SR class.	FQTSS:M	5.4.1.5, 8.6.8.2, 34	Yes [] N/A []
FQTSS3	Support SRP domain boundary port priority regeneration override as defined in 6.9.4, and the default priority regeneration override value defined in Table 6-6, for SR class "B."	FQTSS:M	5.4.1.5, 6.9.4, Table 6-6, 34	Yes [] N/A []
FQTSS4	Support the tables and procedures for mapping priorities to traffic classes as defined in 34.5.	FQTSS:M	5.4.1.5, 34, 34.5	Yes [] N/A []
FQTSS5	Support two or more SR classes (a maximum of seven), and support the operation of the credit-based shaper algorithm on all Ports as the transmission selection algorithm used for those SR classes. The number of SR classes supported shall be stated in the PICS.	FQTSS:O	5.4.1.5, 8.6.8.2, 34.6	Yes [] No [] Number _____
FQTSS6	Support SRP domain boundary port priority regeneration override as defined in 6.9.4, and the default priority regeneration override value defined in Table 6-6, for SR class "A." If more than two SR classes are supported, the default priority regeneration override values used for the additional SR classes shall be stated in the PICS.	FQTSS:O	5.4.1.5, 6.9.4, Table 6-6, 34.6	Yes [] No [] Priority override values _____

A.30 Congestion notification

Item	Feature	Status	References	Support
CN-1	Does the system conform to the required provisions of IEEE Std 802.1AB-2009?	CN: M	item e) in 5.4.3	Yes []
CN-2	Does the system support the use of the Congestion Notification TLV in LLDP?	CN: M	item f) in 5.4.3	Yes []
CN-3	Does the system transmit more than one Congestion Notification TLV in a single LLDPDU?	CN: M		No []
CN-4	Does the system transmit a Congestion Notification TLV with 0 in all of the Per-priority CNPV indicators?	CN: M		No []
CN-5	Does the system implement the Congestion Notification Domain defense variables, procedures, and state machine?	CN: M	32.4, 32.5, 32.6	Yes []
CN-6	Does the Bridge support the creation of at least one CP on at least one Port?	BRG1 AND CN: M	item a) in 5.4.3	Yes []
CN-7	Does the Bridge support the creation of more than one CP on at least one Port?	BRG1 AND CN: O	item i) in 5.4.3	Yes [] No []
CN-8	Does the Bridge support the creation of more than seven CPs on any Port?	BRG1 AND CN: M	item i) in 5.4.3	No []
CN-9	Does every CP on the bridge support all four defense modes separately on each CNPV?	BRG1 AND CN: M	item d) in 5.4.3, 31.1.1, 32.1.1	Yes []
CN-10	Is each CP on the Bridge able to remove CN-TAGs?	BRG1 AND CN: M	item b) in 5.4.3	Yes []
CN-11	Does the PIP perform CNM translation on the return path?	BEB-I AND CN:M	item g) in 5.4.3	Yes []
CN-12	Does the Provider Edge Port perform CNM translation on the return path?	PEB AND CN:M	item h) in 5.4.3	Yes []
CN-13	Is each CP on the system able to generate CNMs?	BRG1 AND CN: M	item b) in 5.4.3	Yes []
CN-14	Does the bridge override the priority of a frame entering a port on a CNPV when in mode cpt-Edge?	BRG1 AND CN: M	32.1.1	Yes []
CN-15	Does the bridge allow any other priority to be remapped to a CNPV when in any mode other than cptDisabled?	BRG1 AND CN: M	32.1.1	No []
CN-16	Do the system's CPs implement the specified variables and procedures?	BRG1 AND CN: M	item a) in 32.7, 32.8, 32.9	Yes []
CN-17	Is the CP's Random() function initialized to a different value each time the system is reset?	BRG1 AND CN: M	32.9.1	Yes []

A.30 Congestion notification (*continued*)

Item	Feature	Status	References	Support
CN-18	Does a system's CP interpret a CN-TAG to any degree beyond simply copying it to the CNM?	BRG1 AND CN: M	33.2.1	No []
CN-19	Does the CP transmit a 0 in the CNM's Version field?	BRG1 AND CN: M	33.4.1	Yes []
CN-20	Does the CP transmit a 0 in the CNM's ReservedV field?	BRG1 AND CN: M	33.4.2	Yes []

A.31 Stream Reservation Protocol

Item	Feature	Status	Reference	Support
	If Stream Reservation Protocol (SRP in Table A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A []
SRP-1	Does the implementation support the exchange of MRPDUs, using the generic MRPDU format defined in 10.8 to exchange MSRP-specific information, as defined in 35.2.2.8.1?	M	10.8, 35.2.2.8.1	Yes []
SRP-2	Is the MSRP Application supported as defined in Clause 35?	M	35	Yes []
SRP-3	Does the implementation propagate registration information in accordance with the operation of MAP for the Base Spanning Tree Context, as specified in 35.2.4?	M	35.2.4	Yes []
SRP-4	Does the implementation forward, filter or discard MAC frames carrying any MRP Application address as the destination MAC address in accordance with the requirements of 8.13.6?	M	8.13.6	Yes []
SRP-5	Is the group MAC Address used as the destination address for MRP-DUs destined for MSRP Participants the group MAC address identified in Table 8-1, Table 8-2, and Table 8-3 as "Individual LAN Scope group address, Nearest Bridge group address"?	M	35.2.2.1, Table 8-1, Table 8-2, Table 8-3	Yes []
SRP-6	Is the EtherType used for MRPDUs destined for MSRP Participants the MSRP EtherType identified in Table 10-2?	M	35.2.2.2, Table 10-2	Yes []

A.31 Stream Reservation Protocol (*continued*)

Item	Feature	Status	Reference	Support	
SRP-7	Does the ProtocolVersion used for the implementation of MSRP take the hexadecimal value 0x00?	M	35.2.2.3	Yes []	
SRP-8	Are the Attribute Type values used in the implementation as specified in 35.2.2.4 and Table 35-1?	M	35.2.2.4, Table 35-1	Yes []	
SRP-9	Are the Attribute Length values used in the implementation as specified in 35.2.2.5 and Table 35-2?	M	35.2.2.5, Table 35-2	Yes []	
SRP-10	Are the MSRP Vector Four-PackedEvents values used in the implementation as specified in 35.2.2.7.2 and Table 35-3?	M	35.2.2.7.2, Table 35-3	Yes []	
SRP-11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 35.2.2.8?	M	35.2.2.8	Yes []	
SRP-12	Does the implementation update Accumulated Latency as the Talker attributes propagate through the Bridge?	M	35.2.2.8.6	Yes []	
SRP-13	Does the implementation update the Failure Information Bridge ID and Code in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.2.8.7	Yes []	
SRP-14	Does the implementation propagate a Talker Advertise as a Talker Failed in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.4.3, Table 35-9	Yes []	
SRP-15	Is talkerPruning and MMRP supported?	O	35.2.4.3.1	Yes []	No []
SRP-16	Are Listener attributes merged and propagated as described in 35.2.4.4?	M	35.2.4.4, Table 35-11, Table 35-14	Yes []	
SRP-17	Does the implementation support updates to the Dynamic Reservation Entries as described in 35.2.4.4.2?	M	8.8 (k), 35.2.4.4.2, Table 35-12	Yes []	
SRP-18	Does the implementation support updates to operIdleSlope as defined in 35.2.4.4.2?	M	35.2.4.4.2, Table 35-13	Yes []	
SRP-19	Does the implementation support the automatic modifications to Stream reservations as described in 35.2.5?	M	35.2.5	Yes []	No []
SRP-20	Are MSRPDUs transmitted on all ports?	M	35.2	Yes []	

A.31 Stream Reservation Protocol (*continued*)

Item	Feature	Status	Reference	Support	
SRP-21	State the number of Talker registrations values that can be supported on each Port.	M	35.2.7	Number _____	
SRP-22	State the number of Listener registrations values that can be supported on each Port.	M	35.2.7	Number _____	
SRP-23	Can the SR_PVID for any Port be assigned the value of the null VLAN ID or VLAN ID FFF (hexadecimal)?	X	35.2.1.4(i), Table 9-2		No []
SRP-24	Does the device support changing the SR_PVID value from the default value shown in Table 9-2?	O	35.2.1.4(i), Table 9-2	Yes []	No []
SRPMDCSN	Does this device support media dependent Coordinated Shared Networking (CSN) functionality on one or more ports?	SRPMDCSN:M OR: SRPMDDOT11:M	Clause C	Yes []	No []
SRPMDCSN	Does this device support media dependent MoCA functionality on one or more ports?	O:1	C.2	Yes []	No []
SRPMDDOT11	Does this device support media dependent IEEE 802.11 Access Point functionality on one or more ports?	O:1	C.3	Yes []	No []
SRPMDCSN-1	Does this device support a single Designated MSRP node (DMN)?	SRPMDCSN:M	35.1.1	Yes []	
SRPMDCMOCA-1	Does this device support DMN Device Attribute Information Element to L2ME message?	SRPMDCMOCA:M	C.2.1.2	Yes []	
SRPMDCMOCA-2	Does this device support DMN selection?	SRPMDCMOCA:M	C.2.1.3	Yes []	
SRPMDCMOCA-3	Does this device support MSRP Attribute Declaration as specified in Table C-1	SRPMDCMOCA:M	C.2.2 Table C-1	Yes []	
SRPMDDOT11-1	Does this device support EDCA-AC?	SRPMDDOT11:M	Table C-5	Yes []	
SRPMDDOT11-2	Is the DMN and the QAP of the IEEE 802.11 BSS co-located in the same device?	SRPMDDOT11:M	C.3.2	Yes []	

A.31 Stream Reservation Protocol (*continued*)

Item	Feature	Status	Reference	Support	
SRPMDDOT11-3	Does the device support MLME primitives specified in Table C-4?	SRPMDDOT11:M	C.3.3, Table C-4	Yes []	
SRPMDDOT11-4	Does the device support VLAN tag encapsulation/de-encapsulation on the 802.11 interface?	SRPMDDOT11:M	C.3.3.1	Yes []	
SRPMDDOT11-5	Is the reservation process an atomic operation?	SRPMDDOT11:M	C.3.1, Figure C-11, Figure C-12, Figure C-13	Yes []	

Annex B

(normative)

PICS proforma—End station implementations⁴⁷

B.1 Introduction

The supplier of a protocol implementation which is claimed to conform to this standard shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use

- a) By the protocol implementor, as a checklist to reduce the risk of failure to conform to the standard through oversight.
- b) By the supplier and acquirer—or potential acquirer—of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis for understanding provided by the standard PICS proforma.
- c) By the user—or potential user—of the implementation, as a basis for initially checking the possibility of interworking with another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSs).
- d) By a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

B.2 Abbreviations and special symbols

B.2.1 Status symbols

M	mandatory
O	optional
O.n	optional, but support of at least one of the group of options labeled by the same numeral n is required
X	prohibited
pred:	conditional-item symbol, including predicate identification: see B.3.4
¬	logical negation, applied to a conditional item's predicate

B.2.2 General abbreviations

N/A	not applicable
PICS	Protocol Implementation Conformance Statement

⁴⁷Copyright 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.

B.3 Instructions for completing the PICS proforma

B.3.1 General structure of the PICS proforma

The first part of the PICS proforma, implementation identification and protocol summary, is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire, divided into several subclauses, each containing a number of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually Yes or No), or by entering a value or a set or range of values. (Note that there are some items where two or more choices from a set of possible answers can apply; all relevant choices are to be marked.)

Each item is identified by an item reference in the first column. The second column contains the question to be answered; the third column records the status of the item—whether support is mandatory, optional, or conditional: see also B.3.4 below. The fourth column contains the reference or references to the material that specifies the item in the main body of this standard, and the fifth column provides the space for the answers.

A supplier may also provide (or be required to provide) further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labelled A_i or X_i , respectively, for cross-referencing purposes, where i is any unambiguous identification for the item (e.g., simply a numeral). There are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformance Statement for the implementation in question.

NOTE—Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation’s configuration capabilities, in case that makes for easier and clearer presentation of the information.

B.3.2 Additional information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or information about aspects of the implementation that are outside the scope of this standard but that have a bearing upon the answers to some items.

References to items of Additional Information may be entered next to any answer in the questionnaire, and may be included in items of Exception Information.

B.3.3 Exception information

It may occasionally happen that a supplier will wish to answer an item with mandatory status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this: instead, the supplier shall write the missing answer into the Support column, together with an X_i reference to an item of Exception Information, and shall provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to this standard.

NOTE—A possible reason for the situation described previously is that a defect in this standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

B.3.4 Conditional status

B.3.4.1 Conditional items

The PICS proforma contains a number of conditional items. These are items for which both the applicability of the item itself, and its status if it does apply—mandatory or optional—are dependent upon whether or not certain other items are supported.

Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the “Not Applicable” answer is selected. Otherwise, individual conditional items are indicated by a conditional symbol in the Status column.

A conditional symbol is of the form “**pred:** S” where **pred** is a predicate as described in B.3.4.2 below, and S is a status symbol, M or O.

If the value of the predicate is true (see B.3.4.2), the conditional item is applicable, and its status is indicated by the status symbol following the predicate: the answer column is to be marked in the usual way. If the value of the predicate is false, the “Not Applicable” (N/A) answer is to be marked.

B.3.4.2 Predicates

A predicate is one of the following:

- a) An item-reference for an item in the PICS proforma: the value of the predicate is true if the item is marked as supported, and is false otherwise.
- b) A predicate-name, for a predicate defined as a Boolean expression constructed by combining item-references using the Boolean operator OR: the value of the predicate is true if one or more of the items is marked as supported.
- c) The logical negation symbol “ \neg ” prefixed to an item-reference or predicate-name: the value of the predicate is true if the value of the predicate formed by omitting the “ \neg ” symbol is false, and vice versa.

Each item whose reference is used in a predicate or predicate definition, or in a preliminary question for grouped conditional items, is indicated by an asterisk in the Item column.

B.3.4.3 References to the text of IEEE Std 802.1D-2004

Many of the tables in the PICS Proforma refer to the text of IEEE Std 802.1D (ANSI/IEEE Std 802.1D). A short form reference, of the form {D}X, is used in the “References” columns of these tables to denote references to clauses, subclauses or tables in IEEE Std 802.1D, where X is the clause, subclause or table identifier.

B.4 PICS proforma for IEEE Std 802.1Q—End station implementations

B.4.1 Implementation identification

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names	

NOTE 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification.

NOTE 2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).

B.4.2 Protocol summary, IEEE Std 802.1Q

Identification of protocol specification	IEEE Std 802.1Q-2011, IEEE Standards for Local and metropolitan area networks—MAC Bridges and Virtual Bridged Local Area Networks	
Identification of amendments and corrigenda to the PICS proforma which have been completed as part of the PICS	Amd. : Corr. : Amd. : Corr. :	
Have any Exception items been required? (See A.3.3: the answer Yes means that the implementation does not conform to IEEE Std 802.1Q)	No []	Yes []

Date of Statement	
-------------------	--

B.5 Major capabilities

Item	Feature	Status	References	Support	
MRPAP	Does the implementation support any MRP applications? If “No” is marked, continue at FQTSSE	O	5.16.1	Yes [] No []	
MMRP	Is the operation of MMRP supported?	O.1	5.16.1, B.6	Yes [] No []	
MVRP	Is automatic configuration and management of VLAN topology using MVRP supported?	O.1	5.16.1, B.7	Yes [] No []	
MSRP	Is the operation of MSRP supported?	O.1	5.16.3, B.10	Yes [] No []	
MRP	Is the Multiple Attribute Registration Protocol (MRP) implemented in support of MRP Applications?	M	10 B.6, B.7, B.8	Yes []	
SPRU	Does the implementation support Source Pruning?	O	5.16, 10.10.3, 11.2.1.1	Yes[] No []	
MRP1	Does the MRP implementation support operation of the Full Participant?	SPRU:O.2 ¬SPRU:O.3	10 B.8	Yes [] No []	
MRP2	Does the MRP implementation support operation of the Full Participant, point-to-point subset?	SPRU:O.2 ¬SPRU:O.3	10 B.8	Yes [] No []	
MRP3	Does the MRP implementation support operation of the Applicant-Only Participant?	¬SPRU:O.3	10 B.8	Yes [] No []	
MRP4	Does the MRP implementation support operation of the Simple-Applicant Participant, point-to-point subset?	¬SPRU:O.3	10 B.8	Yes [] No []	
FQTSSE	Does the implementation support forwarding and queuing for time-sensitive streams?	O	5.18, 34.6	Yes [] No []	
CN	Is congestion notification implemented?	O	5.19, 30, 31, 32, 33	Yes []	No []

B.6 MMRP

Item	Feature	Status	References	Support
	If MMRP is not supported, mark N/A and continue at B.7			N/A []
MMRP1	Does the implementation support the exchange of MMRPDUs, using the generic MRPDU format defined in 10.8 to exchange MMRP-specific information, as defined in 10.12.?	M	5.4.1.3, 10.8, 10.12	Yes []
MMRP2	Is the MMRP Application supported as defined in 10.12?	M	5.4.1.3, 10.12	Yes []
MMRP5	Is the MAP Context Identifier used to identify a VLAN Context equal to the VID used to identify the corresponding VLAN?	M	10.12.1.1	Yes []

B.6 MMRP (continued)

Item	Feature	Status	References	Support
MMRP7	Is the group MAC address used as the destination address for MRPDUs destined for MMRP Participants the group MAC address identified in Table 10-1 as “Customer and Provider Bridge MMRP address”?	M	10.12.1.3	Yes []
MMRP8	Is the EtherType used for MRPDUs destined for MMRP Participants the MMRP EtherType identified in Table 10-2?	M	10.12.1.4, Table 10-2	Yes []
MMRP9	Does the ProtocolVersion used for the implementation of MMRP take the hexadecimal value 0x00?	M	10.12.1.5	Yes []
MMRP10	Are the Attribute Type values used in the implementation as specified in 10.12.1.6?	M	10.12.1.6	Yes []
MMRP11	Does the implementation encode the values in First-Value fields in accordance with the definition in 10.12.1.7?	M	10.12.1.7	Yes []
MMRP12	Is management of the Restricted_MAC_Address_Registration control parameter supported?	O	10.12.2	Yes [] No []
MMRP13	If management of the Restricted_MAC_Address_Registration control parameter is not supported, is the value of this parameter FALSE for all Ports?	¬MMRP12:M	10.12.2	Yes [] N/A []
MMRP14	Does the implementations maintain state information for all attribute values that support the Group service requirement registration?	SPRU:M	10.10.2	Yes [] N/A []
MMRP15	Is the implementation capable of supporting any attribute value in the range of possible values that can be registered using Group membership and individual MAC address registration?	M	10.12.4	Yes []
MMRP16	State the number of Group membership and individual MAC address state values that can be supported on each Port.	M	10.12.4	Number _____

B.7 MVRP

Item	Feature	Status	References	Support
	If MVRP is not supported, mark N/A and continue at B.8			N/A []
MVRP1	Does the implementation support the exchange of MMRPDUs, using the generic MRPDU format defined in 11.2 to exchange MMRP-specific information, as defined in 10.12?	M	5.4.2, 10.8, 11.2	Yes []
MVRP2	Is the MMRP Application supported as defined in 11.2?	M	5.4.2, 11.2	Yes []
MVRP7	Is the group MAC address used as the destination address for MRPDUs destined for MVRP Participants as defined in 11.2.3.1.3?	M	11.2.3.1.3	Yes []
MVRP8	Is the EtherType used for MRPDUs destined for MVRP Participants the MVRP EtherType identified in Table 10-2?	M	11.2.3.1.4, Table 10-2	Yes []
MVRP9	Does the ProtocolVersion used for the implementation of MVRP take the hexadecimal value 0x00?	M	11.2.3.1.5	Yes []

B.7 MVRP (*continued*)

Item	Feature	Status	References	Support
MVRP10	Are the Attribute Type values used in the implementation as specified in 11.2.3.1.6?	M	11.2.3.1.6	Yes []
MVRP11	Does the implementation encode the values in FirstValue fields in accordance with the definition in 11.2.3.1.7?	M	11.2.3.1.7	Yes []
MVRP12	Is the implementation of MVRP capable of supporting all attribute values in the range of possible values that can be registered using MVRP?	SPRU:M	11.2.6	Yes [] N/A []
MVRP13	Is the implementation capable of maintaining current state information for all attributes in the range of possible values?	SPRU:M	11.2.6	Yes [] N/A []

B.8 MRP

Item	Feature	Status	References	Support
MRP5	Does the implementation of MRP meet all of the requirements for interoperability stated in 10.5 that apply to end station operation?	M	10.5	Yes []
MRP6	Does the implementation support the operation of the complete Applicant state machine?	MRP1:M	10.7, 10.7.7	Yes [] N/A []
MRP7	Does the implementation support the operation of the point-to-point subset of the Applicant state machine?	MRP2:M	10.7, 10.7.7	Yes [] N/A []
MRP8	Does the implementation support the operation of the Applicant-Only subset of the Applicant state machine?	MRP3:M	10.7, 10.7.7	Yes [] N/A []
MRP9	Does the implementation support the operation of the Simple-Applicant subset of the Applicant state machine?	MRP4:M	10.7, 10.7.7	Yes [] N/A []
MRP10	Does the implementation support the operation of the Registrar state machine?	MRP1:M MRP2:M	10.7, 10.7.8	Yes [] N/A []
MRP11	Does the implementation support the operation of the LeaveAll state machine?	MRP1:M MRP2:M	10.7, 10.7.9	Yes [] N/A []
MRP12	Does the implementation support the operation of the PeriodicTransmission state machine?	M	10.7, 10.7.10	Yes []

B.9 Forwarding and queuing for time-sensitive streams

Item	Feature	Status	References	Support
FQT SSE1	Support a minimum of two traffic classes, of which one supports the strict priority algorithm and the other is an SR class?	FQT SSE:M	5.18, 8.6.8.1, 34.6	Yes [] N/A []
FQT SSE2	Support the credit-based shaper algorithm as the transmission selection algorithm for frames transmitted for each stream associated with the SR class.	FQT SSE:M	5.18, 8.6.8.2, 34.6	Yes [] N/A []
FQT SSE3	Support the operation of the credit-based shaper algorithm on all Ports as the transmission selection algorithm used for the SR class.	FQT SSE:M	5.18, 8.6.8.2, 34.6	Yes [] N/A []
FQT SSE4	Use the default priority associated with SR class “B” as shown in Table 6-6 as the priority value carried in transmitted SR class “B” data frames.	FQT SSE:M	5.18, Table 6-6, 34.6	Yes [] N/A []
FQT SSE5	Support two or more SR classes (a maximum of seven), and support the operation of the credit-based shaper algorithm on all Ports as the transmission selection algorithm used for those SR classes. The number of SR classes supported shall be stated in the PICS.	FQT SSE:O	5.18, 8.6.8.2, 34.6	Yes [] No [] Number _____
FQT SSE6	Use the default priority associated with SR class “A” as shown in Table 6-6 as the priority value carried in transmitted SR class “A” data frames. If more than two SR classes are supported, the priority value carried in transmitted data frames for the additional SR classes shall be stated in the PICS.	FQT SSE:O	5.18, Table 6-6, 34.6	Yes [] No [] Priority over- ride values _____

B.10 SRP (Stream Reservation Protocol)

Item	Feature	Status	Reference	Support
	If SRP is not supported, mark N/A and ignore the remainder of this table.			N/A []
SRP-1	Does the implementation support the exchange of MRPDUs, using the generic MRPDU format defined in 10.8 to exchange MSRP-specific information, as defined in 35.2.2.8.1?	M	5.4.4, 10.8, 35.2.2.8.1	Yes []
SRP-2	Is the MSRP Application supported as defined in Clause 35?	M	5.4.4, 35	Yes []
SRP-3	Is the group MAC Address used as the destination address for MRPDUs destined for MSRP Participants the group MAC address identified in Table 8-1, Table 8-2, Table 8-3 as “Individual LAN Scope group address, Nearest Bridge group address”?	M	35.2.2.1, Table 8-1, Table 8-2, Table 8-3	Yes []
SRP-4	Is the EtherType used for MRPDUs destined for MSRP Participants the MSRP EtherType identified in Table 10-2?	M	35.2.2.2, Table 10-2	Yes []

B.10 SRP (Stream Reservation Protocol) (continued)

Item	Feature	Status	Reference	Support	
SRP-5	Does the ProtocolVersion used for the implementation of MSRP take the hexadecimal value 0x00?	M	35.2.2.3	Yes []	
SRP-6	Are the Attribute Type values used in the implementation as specified in 35.2.2.4 and Table 35-1?	M	35.2.2.4, Table 35-1	Yes []	
SRP-7	Are the Attribute Length values used in the implementation as specified in 35.2.2.5 and Table 35-2?	M	35.2.2.5, Table 35-2	Yes []	
SRP-8	Are the MSRP Vector FourPackedEvents values used in the implementation as specified in 35.2.2.7.2 and Table 35-3?	M	35.2.2.7.2, Table 35-3	Yes []	
SRP-9	Does the implementation encode the values in FirstValue fields in accordance with the definition in 35.2.2.8?	M	35.2.2.8	Yes []	
SRP-10	Does the Talker implementation populate the Accumulated Latency with a reasonable, nonzero value?	M	35.2.2.8.6	Yes []	
SRP-11	Does the implementation update the Failure Information Bridge ID and Code in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.2.8.7	Yes []	
SRP-12	Does the implementaiton create a Talker Failed in the event of insufficient bandwidth or resources through a Bridge?	M	35.2.4.3, 35-10	Yes []	
SRP-13	Is talkerPruning and MMRP supported?	O	35.2.4.3.1	Yes []	No []
SRP-14	Are MSRPDUs transmitted on all ports?	M	35.2	Yes []	
SRP-15	State the number of Talker registration values that can be supported on each Port.	M	35.2.7	Number _____	
SRP-16	State the number of Listener registration values that can be supported on each Port.	M	35.2.7	Number _____	
SRP-17	Does the Listener issue an appropriate MVRP VLAN membership request before attaching to a Stream?	M	35.1.2.2	Yes []	
SRP-18	When MAC_Operational transitions to TRUE does the device declare the default SR class priority value as the SRclassPriority prior to receving an SRclassPriority declaration from its neighbor?	M	6.6.2, 35.2.2.9.3, Table 6-6	Yes []	

B.10 SRP (Stream Reservation Protocol) (*continued*)

Item	Feature	Status	Reference	Support	
SPR-19	Does the device set SRclassPriority to the value declared by the neighboring device?	M	35.2.2.9.3	Yes []	
SRP-20	When MAC_Operational transitions to TRUE does the device declare the default SR_PVID value as the SRclassVID prior to receiving an SRclassVID declaration from its neighbor?	M	6.6.2, 35.2.2.9.4, Table 9-2	Yes []	
SPR-21	Does the device set SRclassVID to the value declared by the neighboring device?	M	35.2.2.9.4	Yes []	
SRPMDCSN	Does this device support media dependent Coordinated Shared Networking (CSN) functionality on one or more ports?	SRPMDDMOCA:M OR: SRPMDDOT11:M	C	Yes []	No []
SRPMDDMOCA	Does this device support media dependent MoCA functionality on one or more ports?	O:1	C.2	Yes []	No []
SRPMDDOT11	Does this device support media dependent IEEE 802.11 Access Point functionality on one or more ports?	O:1	C.3	Yes []	No []
SRPMDCSN-1	Does this device support a single Designated MSRP node (DMN)?	SRPMDCSN:M	35.1.1	Yes []	
SRPMDDMOCA-1	Does this device support DMN Device Attribute Information Element to L2ME message?	SRPMDDMOCA:M	C.2.1.2	Yes []	
SRPMDDMOCA-2	Does this device support DMN selection?	SRPMDDMOCA:M	C.2.1.3	Yes []	
SRPMDDMOCA-3	Does this device support MSRP Attribute Declaration as specified in Table C-1?	SRPMDDMOCA:M	C.2.2 Table C-1	Yes []	
SRPMDDOT11-1	Does this device support EDCA-AC?	SRPMDDOT11:M	Table C-5	Yes []	
SRPMDDOT11-2	Is the DMN and the QAP of the IEEE 802.11 BSS co-located in the same device?	SRPMDDOT11:M	C.3.2	Yes []	
SRPMDDOT11-3	Does the device support MLME primitives specified in Table C-4?	SRPMDDOT11:M	C.3.3, Table C-4	Yes []	
SRPMDDOT11-4	Does the device support VLAN tag encapsulation/de-encapsulation on the 802.11 interface?	SRPMDDOT11:M	C.3.3.1	Yes []	
SRPMDDOT11-5	Is the reservation process an atomic operation?	SRPMDDOT11:M	C.3.1, Figure C-11, Figure C-12, Figure C-13	Yes []	

B.11 Congestion notification

Item	Feature	Status	References	Support
CN-1	Does the system conform to the required provisions of IEEE Std 802.1AB-2009?	CN: M	5.19	Yes []
CN-2	Does the system support the use of the Congestion Notification TLV in LLDP?	CN: M	5.19	Yes []
CN-3	Does the system transmit more than one Congestion Notification TLV in a single LLDPDU?	CN: M		No []
CN-4	Does the system transmit a Congestion Notification TLV with 0 in all of the Per-priority CNPV indicators?	CN: M		No []
CN-5	Does the system implement the Congestion Notification Domain defense variables, procedures, and state machine?	CN: M	32.4, 32.5, 32.6	Yes []
CN-6	Does the end station support the creation of at least one RP?	CN: M	5.19	Yes []
CN-7	Does the end station allow more than rpppMaxRps RPs to be created on one port and priority?	CN: M	32.10.1	No []
CN-8	Does the end station support at least the cptDisabled and cptInterior defense modes separately on each CNPV?	CN: M	5.19, 32.1.1	Yes []
CN-9	Does the end station add a tag to frames transmitted from CCFs?	CN: O	30.5, 31.2.2.5	Yes [] No []
CN-10	Does the end station support both the cptInterior, and cptInteriorReady defense modes separately on each CNPV?	CN: O	5.19, 31.2.2.5, 32.14.3	Yes [] No []
CN-11	Does the end station support the cptEdge defense mode on any CNPV?	CN: O	5.19, 32.4.9	Yes [] No []
CN-12	Does the end station accept CN-TAGs in data frames on any CNPV?	CN: O	5.19, 32.4.10	Yes [] No []
CN-13	Does the end station support the creation of more than one RP?	CN: O	5.19, 31.2.1	Yes [] No []
CN-14	Reserved			
CN-15	Do the end station's RP(s) limit the output frame rate in response to CNMs received?	CN: M	5.19	Yes []
CN-16	Does the end station distinguish correctly to which RP a given CNM is directed?	CN-13: M	5.19	Yes []
CN-17	Does the end station support the creation of at least one CP?	CN: O	5.19	Yes [] No []
CN-18	Does the end station's have a single flow queue as its default configuration?	CN: M	31.2.1	Yes []
CN-19	Does the end station change its method for assigning frames to Flow queues in a manner that impairs the ability of compliant CPs to throttle its flows?	CN: M	31.2.1	No []
CN-20	Does the end station change its method for assigning frames to Flow queues in a manner that significantly increases the likelihood of frame misordering, and thus impacts higher layer functions?	CN: M	31.2.1	No []
CN-21	Does the end station's Rate Limiters always meet the specified formula for L_T ?	CN: M	31.2.2.4 equation (1)	Yes []

B.11 Congestion notification (*continued*)

Item	Feature	Status	References	Support
CN-22	Can the end station's Flow Selection functions drop frames between the Flow queue and the Flow multiplexer?	CN: M	31.2.2.5	No []
CN-23	Does the end station output frames with CN-TAGs on a priority operating in mode cptEdge or cptInterior?	CN: M	32.1.1	No []
CN-24	Do the end station's RPs implement the specified timers, variables, procedures, and state machines?	CN: M	item b) in 32.7, 32.12, 32.13, 32.14, 32.15	Yes []
CN-25	Does the end station insert the CN-TAG and VLAN-tag in the correct order (addresses, VLAN-tag, CN-TAG)?	CN: M	33.2	Yes []
CN-26	Does the end station ignore the CNM's Version field?	CN: M	33.4.1	Yes []
CN-27	Does the end station ignore the CNM's ReservedV field?	CN: M	33.4.2	Yes []
CN-28	Does the end station assign meaning to subfields within a received CNM's Congestion Point Identifier field?	CN: M	33.4.4	No []
CN-29	Does the end station ignore received CNMs that are too short?	CN: M	item a) in 33.4.11	Yes []
CN-30	Does the end station ignore received CNMs that have no CN-TAG?	CN: O	item b) in 33.4.11	Yes [] No []
CN-31	Does the end station ignore received CNMs that have nonzero values in the Version or ReservedV fields?	CN: M	item c) in 33.4.11	No []
CN-32	Does the end station support all of the objects required to manage its RP(s)?	CN: M	5.19, 12.21.1, 12.21.6	Yes []
CN-33	Does the end station use the same Reaction Point group managed object to manage RPs serving more than one CNPV?	CN: M	32.11	No []
CN-34	Does the end station support all of the objects in the CN component managed object?	CN-17: M	5.19, 12.21.1	Yes []
CN-35	Does the end station support all of the objects in the CN component priority managed object?	CN-17: M	5.19, 12.21.2	Yes []
CN-36	Does the end station support all of the objects in the CN Port priority managed object?	CN-17: M	5.19, 12.21.3	Yes []
CN-37	Does the end station support all of the objects in the Congestion Point managed object?	CN-17: M	5.19, 12.21.4	Yes []

Annex C

(normative)

DMN (Designated MSRP Node) Implementations

This annex describes the DMN implementation on an IEEE 802.11 Network and Coordinated Shared Networks (CSNs)

C.1 Designated MSRP nodes on CSNs

A CSN is a contention-free, time-division multiplexed-access network, supporting reserved bandwidth based on priority or flow (QoS). One of the nodes of the CSN acts as the Network Coordinator (NC) node, granting transmission opportunities to the other nodes of the network. The NC node also acts as the bandwidth resource manager of the network.

C.1.1 Coordinated Shared Network (CSN) characteristics

CSNs support two types of transmissions: unicast transmission for node-to-node transmission and multicast/broadcast transmission for one-node-to-other/all-nodes transmission. Each node-to-node link has its own bandwidth characteristics that could change over time due to the periodic ranging of the link. The multicast/broadcast transmission characteristics are the lowest common characteristics of multiple/all the links of the network.

A CSN network is physically a shared network, in that a CSN node has a single physical port connected to the half-duplex medium, but is also a logically fully-connected one-hop mesh network, in that every node could transmit to every other node using its own profile over the shared medium.

Figure C-1 illustrates a CSN network acting as a backbone interconnecting AV systems.

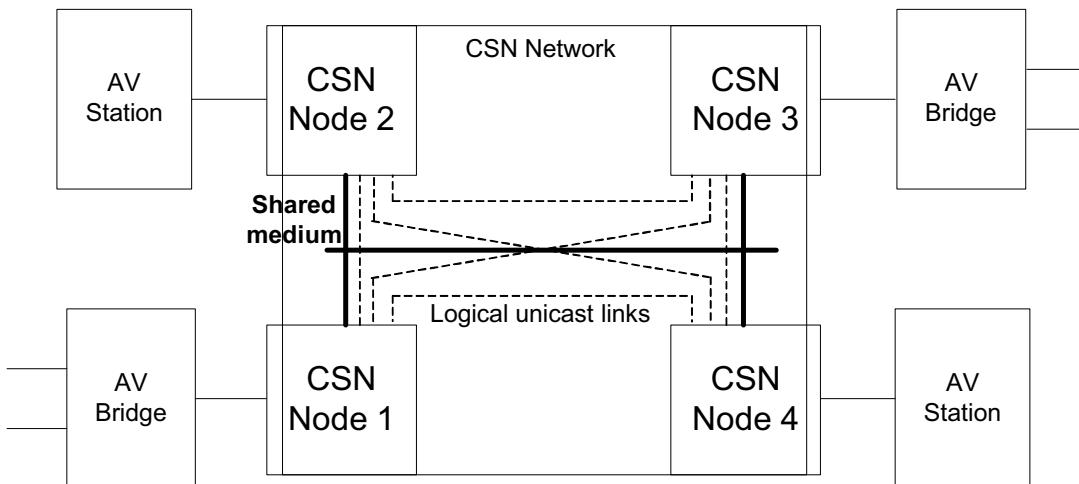


Figure C-1—CSN Backbone

Depending on the CSN technology, the Network Coordinator node either may be a fixed node or may be dynamically selected during normal operation.

C.1.2 Designated MSRP Node handling on CSN

From the bandwidth reservation stand point a CSN network is modeled as a Bridge as illustrated by Figure C-2. Each node-to-node link is equivalent to a Bridge's path from an ingress port to an egress port.

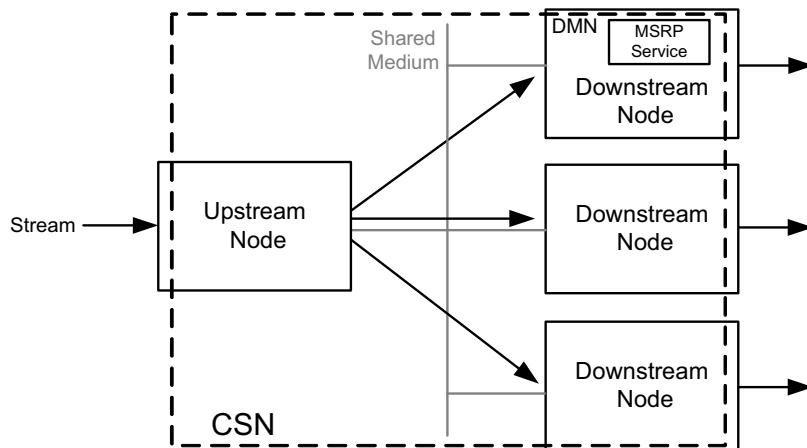


Figure C-2—Bridge's CSN Model for Bandwidth Reservation

A CSN shall provide a single entity called the Designated MSRP Node or DMN (35.1.1) which communicates with the MSRP Service to manage the CSN bandwidth resources for the MSRP streams.

NOTE 1—The MSRP Service supplies both MSRP Attribute Propagation (MAP) for the CSN and MSRP Attribute Distribution (MAD) for each MSRP-aware CSN node.

NOTE 2—An end station node provides an end station MSRP implementation as outlined in 5.16.3. This MSRP implementation is logically on the upper layer side of this interface. This standard expects that an MSRP-aware CSN node will have a single non-CSN interface. If more interfaces are present they are expected to be connected through a bridge function. If this node happens to be the DMN then this end station MSRP implementation is separate and distinct from the DMN's MSRP Service.

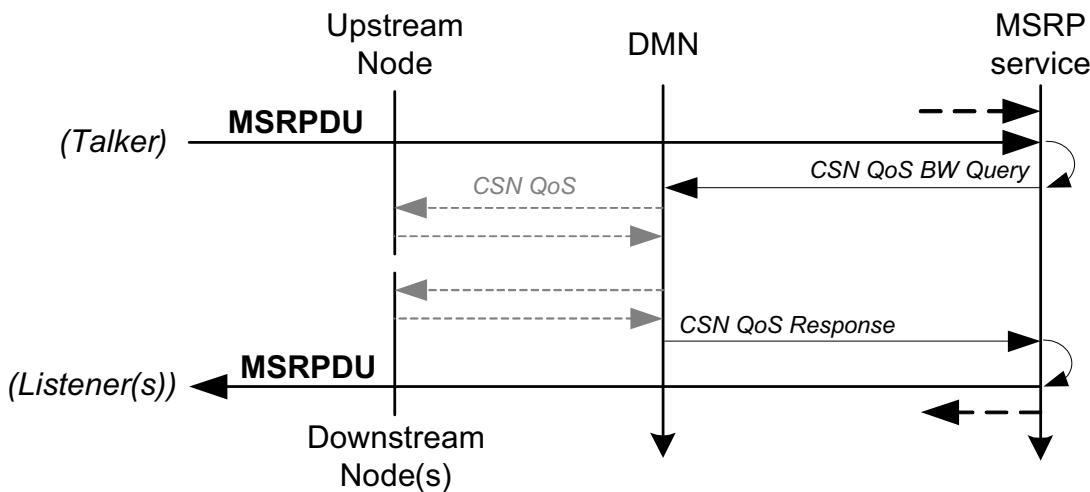
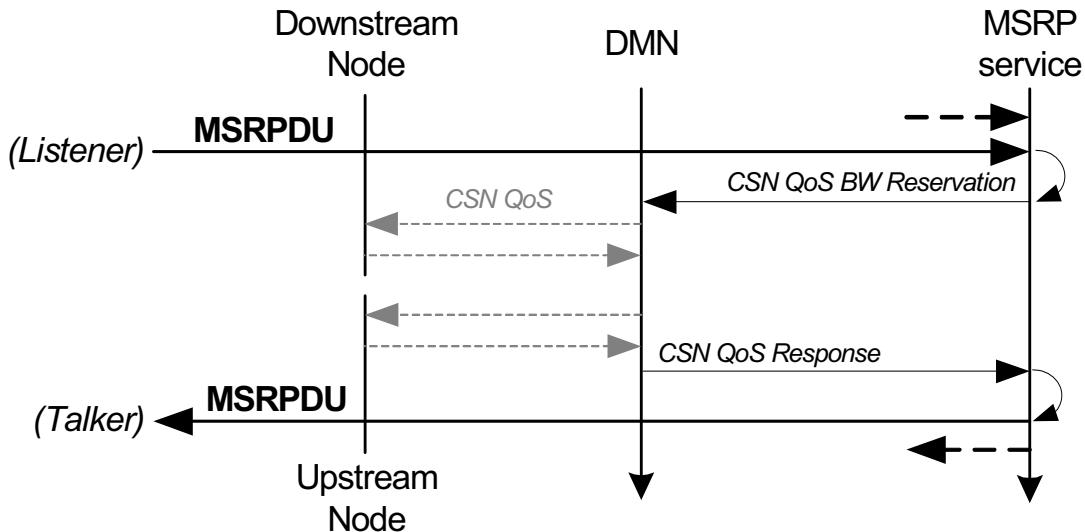
C.1.2.1 DMN Selection and Migration

Depending on the CSN technology, the DMN might correspond to a static node or dynamically migrate between nodes during normal operation. The DMN selection is network specific and described in C.2.1 and C.3.2.

Over time the DMN constructs its database by handling the MSRP Talker and Listener Declarations generated by the nodes of the CSN. If the DMN migrates, the new DMN broadcasts an MRP LeaveAll message to all the nodes of the CSN which will force its neighbors to redeclare their attributes. The MSRP Participant nodes answer the MRP LeaveAll message by sending an MRP JoinIn message consumed by the DMN as an MRP Re-Declare! message. These redeclarations permit the new DMN to immediately build its database.

C.1.3 MSRPDU handling on a CSN

Figure C-3 and Figure C-4 illustrate the flow and handling of MSRPDU messages on a CSN.

**Figure C-3—Talker MSRPDU flow****Figure C-4—Listener MSRPDU flow**

- A MSRP-aware CSN node identifies MSRPDUs received on its non-CSN interface (the interface either to another network media or to upper layers of the node) by their destination MAC address (35.2.2.1) and EtherType (35.2.2.2).
- Non-DMN nodes send MSRPDUs to the DMN over the CSN.
- The DMN delivers MSRPDUs, along with information about the originating interface, to the MSRP Service.
- The DMN translates the MSRP TSpec parameters into CSN QoS parameters and invokes the CSN's Protocol Specific QoS transactions with the CSN Network Coordinator (C.2.2, C.3.3) as follows:
 - When the DMN receives a Talker Advertise message originated from an upstream CSN node, the DMN invokes a bandwidth query transaction with the CSN Network Coordinator to check whether or not the bandwidth advertised in the message's TSpec is available on each upstream

to downstream node link of the CSN network. In addition the DMN associates the MSRP attributes with the CSN's native QoS records via the StreamID (which is included in the CSN frames).

- 2) When the DMN receives a Listener Ready message originated from a downstream CSN node, the DMN invokes a bandwidth reservation transaction with the CSN QoS manager to reserve the bandwidth associated with the message's StreamID on the downstream to upstream CSN node link.
- e) After the DMN completes the CSN QoS transactions, the DMN behaves as an MSRP application on a Bridge and propagates (MAP) and distributes (MAD) MSRP attributes (35.2).

NOTE—When the MAD function of the MSRP Service distributes attributes it will deliver MSRP DUs to the DMN's non-CSN interface and/or send them to the appropriate MSRP-aware CSN nodes.

C.1.4 CSN bandwidth fluctuations

Bandwidth on a CSN may fluctuate depending on interference experienced by the media. The MSRP Attribute Propagation (35.2.4) supports a *bandwidthAvailabilityChanged* notification when bandwidth increases or decreases across the media. When a bandwidth changed is encountered, MAP will reassess reservation requests to maintain appropriate bandwidth utilization.

C.2 Designated MSRP Node on MoCA

NOTE—The discussion that follows is based on terminology found in the MoCA Specifications [B42], [B43], and [B44].

C.2.1 DMN Selection on MoCA Network

C.2.1.1 DMN capable node discovery

A DMN capable node shall append the *IEEE DMN Device Attribute Information Element* (C.2.1.2) to the L2ME payload of the Device Discovery Protocol SUBMIT L2ME transaction message specified in the MoCA v2.0 Specification [B44].

Upon completion of the L2ME Device Discovery transaction all the DMN-capable nodes of the MoCA network share the same information as follows:

- a) which MoCA nodes are DMN capable,
- b) which MoCA node is selected as the DMN.

If no DMN is selected, the DMN selection shall be performed (C.2.1.3).

C.2.1.2 IEEE DMN Device Attribute IE

C.2.1.2.1 General

The fields of the IEEE DMN Device Attribute IE are specified in Figure C-5 and C.2.1.2.2 through C.2.1.2.7. The general format of the Device Attribute Information Element is described in the MoCA v2.0 Specification [B44].

Bits	Octets	Offset From Start of IE
8 7 6 5 4 3 2 1		
ATTRIBUTE_ID	1	0
LENGTH	1	1
----- VENDOR_ID -----	2	2
TLV_TYPE	1	4
TLV_LENGTH	1	5
----- TLV_VALUE -----	2	6

Figure C-5—IEEE DMN Device Attribute IE**C.2.1.2.2 ATTRIBUTE_ID (Enumeration8)**

The value of the ATTRIBUTE_ID is 0xFF.

C.2.1.2.3 LENGTH (UInteger8)

The value of the LENGTH is 1.

NOTE—The actual length of the Attribute IE in bits is (LENGTH + 1) × 32.

C.2.1.2.4 VENDOR_ID (Enumeration16)

The value of VENDOR_ID is 0x0090 (IEEE 802.1 AVB).

C.2.1.2.5 TLV_TYPE (Enumeration16)

The value of the TLV_TYPE is 1 (SRP).

C.2.1.2.6 TLV_LENGTH (UInteger8)

The length of the TLV in octets is 4.

C.2.1.2.7 TLV_VALUE (Octet2)

The meaning of the field of the TLV_VALUE are specified as described in C.2.1.2.7.1 through C.2.1.2.7.3.

C.2.1.2.7.1 DMNcapable (Bit 0 - Boolean)

A value of 1 indicates the node is capable of acting as the DMN of the network. A value of 0 indicates the node is not capable to act as a DMN.

C.2.1.2.7.2 DMNselected (Bit 1 - Boolean)

A value of 1 indicates the node has been selected as the DMN of the network. A value of 0 indicates the node is not the selected DMN.

C.2.1.2.7.3 Reserved (Bit 2..15)

These bits are reserved for future usage.

C.2.1.3 DMN selection and confirmation

If either 1) the *NODE_BITMASK* field into MAP frames indicates that the selected DMN has been removed from the network (due to failure, power state/down, etc.) or 2) the DMN node discovery (C.2.1.1) does not indicate a DMN selected node, the DMN capable node with the lowest node ID will start acting as the DMN and confirm the selection to the other DMN capable nodes by generating a L2ME DMN Confirmation Transaction (C.2.1.3.1).

NOTE—"MAP frame" is defined in the MoCA v1.0 Specification [B42].

C.2.1.3.1 L2ME DMN Confirmation Transaction

C.2.1.3.1.1 Overview of DMN Confirmation Transaction

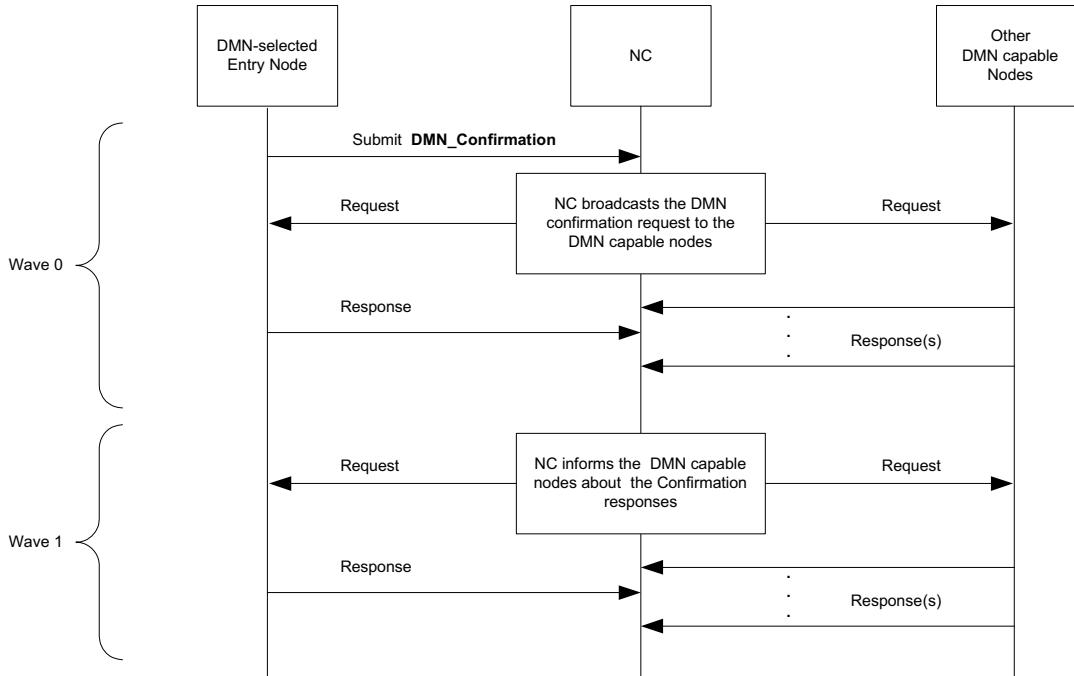
Figure C-6 provides an overview of the signals exchanged among the nodes during an L2ME DMN Confirmation Transaction. As shown in the figure, the NC node starts the transaction either when it receives a Submit L2ME Frame from a node (called the DMN Selected Entry Node for that Transaction) or on its own. The transaction includes two L2ME Waves. The details of each message exchanged during the DMN Confirmation Transaction are provided below.

C.2.1.3.1.2 DMN Confirmation Submit

The DMN Confirmation Transaction starts when the DMN Selected Entry Node sends a Submit L2ME Frame to the NC.

The general fields of the Submit L2ME frame are as specified in Section 2.2.3.1 of the MoCA v1.1 Specification [B43]. The parameters in the Submit L2ME Frame are set as follows:

- a) VENDOR_ID (see C.2.1.2.4)
- b) TRANS_TYPE = 0x1 (SRP)
- c) TRANS_SUBTYPE = 0x1 (DMN Confirmation)
- d) WAVE0_NODEMASK = (DMN capable nodes)
- e) MSG_PRIORITY = 0xF0
- f) TXN_LAST_WAVE_NUM = 0x1
- g) L2ME_PAYLOAD = 0 bytes

**Figure C-6—DMN Confirmation Transaction****C.2.1.3.1.3 Request L2ME Frame of Wave 0 of DMN Confirmation Transaction**

In the Wave 0 that follows the DMN Confirmation Submit message, the NC node initiates a Request L2ME Frame based on the Submit L2ME Frame.

The general fields of the Request L2ME frame are as specified in the in Section 2.2.3.2 of the MoCA v1.1 Specification [B43].

C.2.1.3.1.4 Response L2ME Frame of Wave 0 of DMN Confirmation Transaction

Each of the requested DMN capable nodes sends a Response L2ME Frame to acknowledge the confirmation request.

The general fields of the Response L2ME frame are as specified in Section 2.2.3.3 of the MoCA v1.1 Specification [B43]. The parameters in the Response L2ME Frame are set as follows:

- RESP_STATUS <INTERPRETED> = '1'
- RESP_STATUS <IN_NEXT_WAVE> = '1'
- L2ME_PAYLOAD = 0 bytes

C.2.1.3.1.5 Request L2ME Frame of Wave 1 of DMN Confirmation Transaction

In Wave 1, the NC node informs the DMN capable nodes about the acknowledgement results from Wave 0. The NC node initiates Wave 1 using a Request L2ME Frame with the “concatenated” type of L2ME_PAYLOAD.

The general fields of the Request L2ME frame are as specified in Section 2.2.3.1 of the MoCA v1.1 Specification [B43]. The “concatenated” L2ME_Payload is as specified in Table 2-4 of the MoCA v1.1 Specification [B43].

C.2.1.3.1.6 Response L2ME Frame of Wave 1 of DMN Confirmation Transaction

The DMN Confirmation Transaction is completed when the DMN Selected Entry node and the other DMN capable nodes send their final Response L2ME Frame to the NC node.

The general fields of the Response L2ME frame are as specified in Section 2.2.3.3 of the MoCA v1.1 Specification [B43]. The parameters in the Response L2ME Frame are set as follows:

- a) RESP_STATUS <INTERPRETED> = '1'
- b) RESP_STATUS <IN_NEXT_WAVE> = '0'
- c) L2ME_PAYLOAD = 0 bytes

C.2.2 MoCA network bandwidth management

The MSRP service within the MoCA network manages the MoCA bandwidth for the MSRP streams by invoking the MoCA native PQoS transactions. The DMN shall map the MSRP Attribute Declaration and the resultant MAD declarations as described in Table C-1.

Table C-1—SRP to MoCA PQoS Transaction mapping

MSRP Attribute	MAD Primitive	MoCA PQoS Transactions	Description
Talker Advertise	MAD_Join.request(new)	Create PQoS Flow	<p>Query bandwidth without reservation</p> <p>NOTE—MoCA PQoS APIs do not include a Bandwidth Query specific API. Therefore, bandwidth is queried by invoking a CreatePQoSFlow reservation for more bandwidth than the network could provide. The request will fail and CreatePQoSFlow will then return a failure status that includes the bandwidth available for reservations.</p>
Listener Ready or Listener Ready Failed	MAD_Join.request(new)	Create PQoS Flow	Reserve bandwidth for a stream
Listener Ready or Listener Ready Failed	MAD_Join.request()	Update PQoS Flow	Renew the bandwidth reservation (leased time) for a stream
Talker or Listener Leave	MAD_Leave.request()	Delete PQoS Flow	Free bandwidth associated with a stream

Table C-2 describes the mapping between SRP TSpec components and MoCA PQoS TSPEC parameters.

Table C-3 describes the mapping of the SRP StreamID to MoCA PQoS Flow transactions. The MoCA Flow parameters includes a 32-bit field that allows an application, like SRP, to store application specific information. SRP will use this field (FLOW_TAG) to store the 16-bit unique ID portion of the StreamID.

Table C-2—SRP TSpec to MoCA TSPEC mapping

SRP TSpec	MoCA PQoS TSPEC
MaxFrameSize	Max Packet Size
MaxFrameSize × MaxIntervalFrames	Peak Data Rate
MaxFrameSize × MaxIntervalFrames × Class B class measurement interval (34.4)	(Max) Burst Size

Table C-3—SRP StreamID to MoCA PQoS Flow transaction mapping

SRP StreamID	MoCA PQoS Flow Transaction		
	Transaction	L2ME Payload	Field
EUI-48 MAC Address	Create PQoS Flow Update PQoS Flow	Submit	PACKET_DA
	Query PQoS Flow	Response	
16-bit Unique ID	Create PQoS Flow Update PQoS Flow	Submit	FLOW_TAG
	Query PQoS Flow	Response	

C.3 Designated MSRP Nodes on IEEE 802.11 media

NOTE—Even though IEEE Std 802.11 is not a CSN, it uses the same DMN concepts described below.

From the bandwidth reservation standpoint an IEEE 802.11 BSS network is modeled as a Bridge as illustrated by Figure C-7, Figure C-8, and Figure C-9. Each STA-AP link, STA-AP-STA link and optional STA-STA DLS direct link is equivalent to the path from an input to an output Bridge's port.

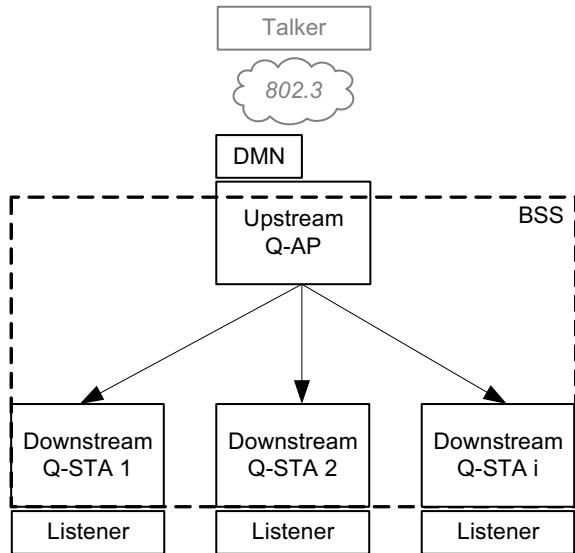
An IEEE 802.11 BSS provides a single entity called the Designated MSRP Node (DMN) (35.1.1) to manage the BSS bandwidth resources for the MSRP streams.

C.3.1 MSRP Handling

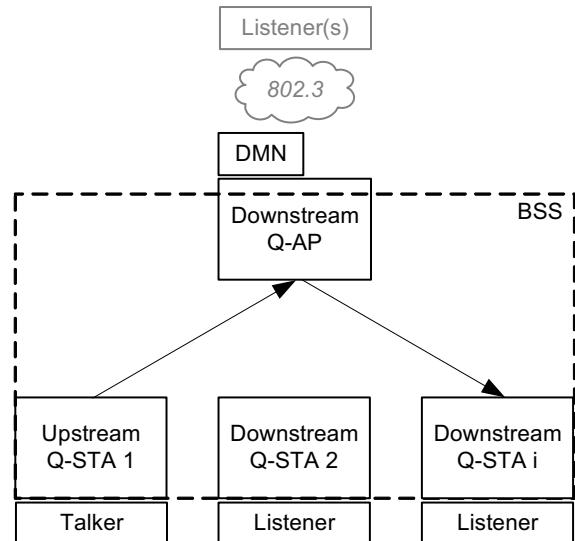
MSRPDUs are transparently transported by the IEEE 802.11 BSS network and delivered to the DMN.

The DMN maps the MSRP commands into IEEE 802.11 MLME TS commands and interacts with the AP through the AP's MLME SAP.

Figure C-10 through Figure C-13 describe the flow of information between the MSRP and IEEE 802.11 entities, and corresponding over the air IEEE 802.11 frames. Figure C-10 is an example of the IEEE 802.11 bandwidth query process associated with an MSRP Talker Advertise.



**Figure C-7—Bandwidth Reservation - Bridge Model for IEEE 802.11 BSS
(STA Downstream Port)**



**Figure C-8—Bandwidth Reservation - Bridge Model for IEEE 802.11 BSS
(STA Upstream Port)**

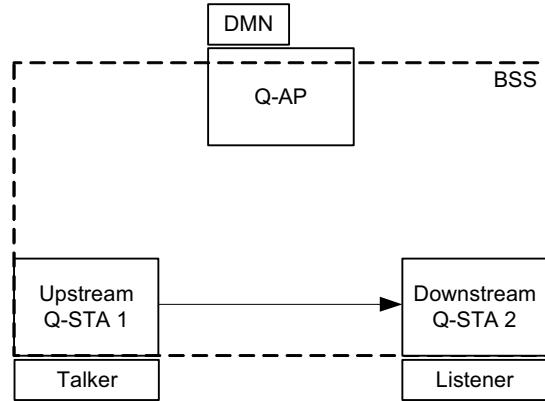


Figure C-9—Bandwidth Reservation - Bridge Model for IEEE 802.11 BSS (Direct Link Setup)

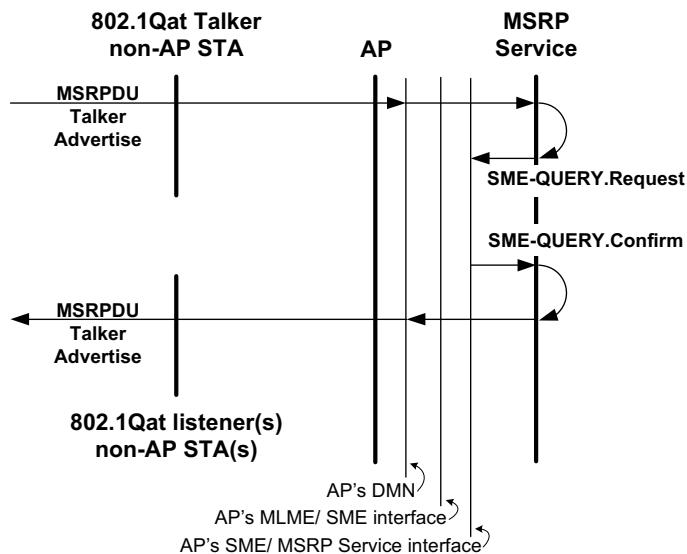


Figure C-10—MSRP/802.11 Query Flows

Figure C-10 is an example of the reservation process associated with a Listener non-AP STA requesting a Stream from the Talker non-AP STA. The diagram at the left of the figure shows the Listener non-AP STA sending a Listener Ready (A) through the AP's MSRP Service, which then propagates that Listener Ready (B) to the Talker non-AP STA. The message flow on the right of the figure shows the corresponding IEEE 802.11 message exchanges associated with the two Listener Readys (A & B).

There are two reservations (A & B) required to allow the Stream to flow from Talker to Listener. In Figure C-11, Figure C-12 and Figure C-13 the reservation process must be an atomic operation so if either reservation fails then both reservations shall be removed and the MSRP attributes updated accordingly.

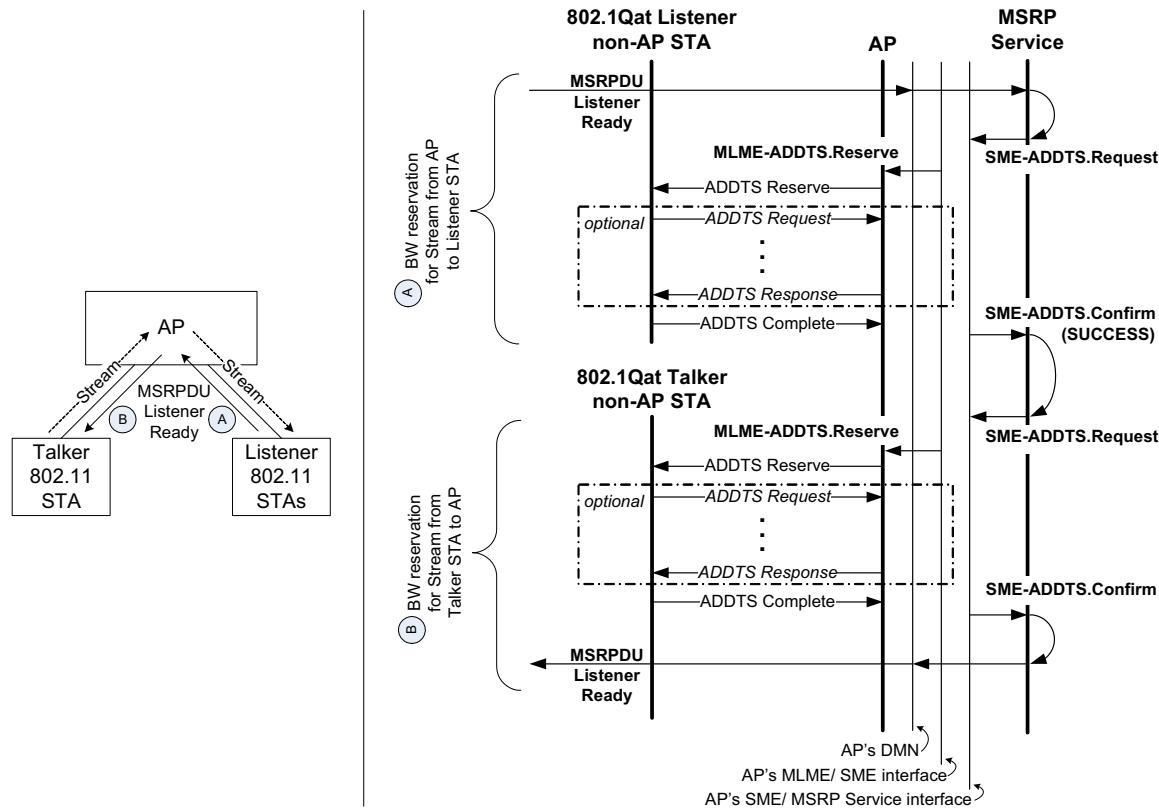


Figure C-11—MSRP/802.11 Talker STA to Listener STA Reservation Flows

Figure C-12 is an example of the reservation process associated with a “Bridged” Listener requesting a Stream from a Talker non-AP STA. The Listener sends a Listener Ready (A) from the “Bridged” side of the network and bandwidth is reserved as explained in SRP (Clause 35). The IEEE 802.11 message exchanges associated with the Listener Ready (B) propagating to the Talker non-AP STA are also shown.

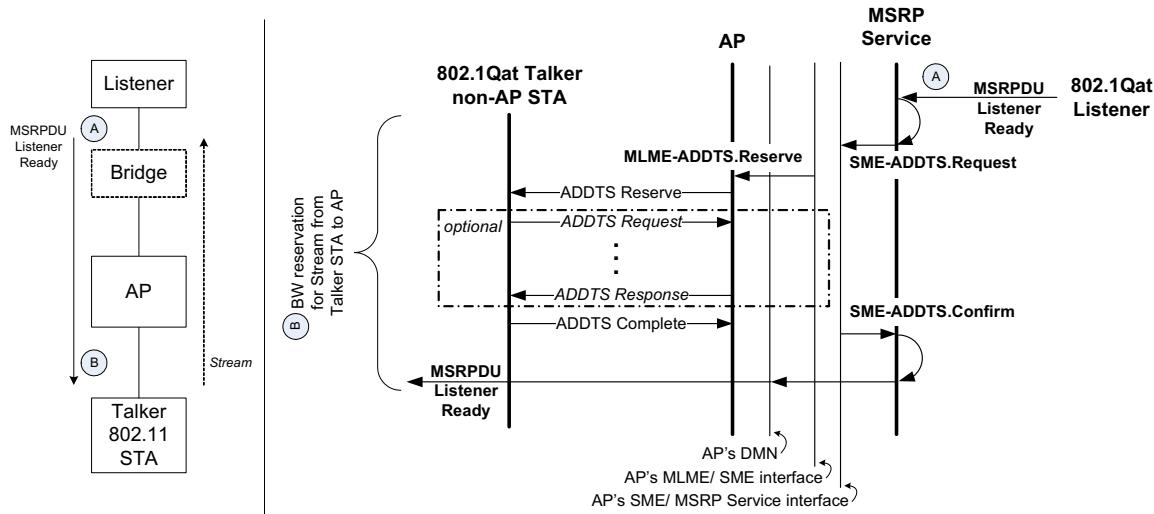


Figure C-12—MSRP/802.11 “Bridged” Listener to Talker STA Reservation Flows

Figure C-13 is an example of the reservation process associated with a Listener non-AP STA requesting a Stream from a “Bridged” Talker. The Listener non-AP STA sends a Listener Ready (A) through the AP’s MSRP Service, resulting in the IEEE 802.11 message exchanges shown. The reservation process associated with the Listener Ready (B) sent to the “Bridged” Talker is explained in SRP (Clause 35).

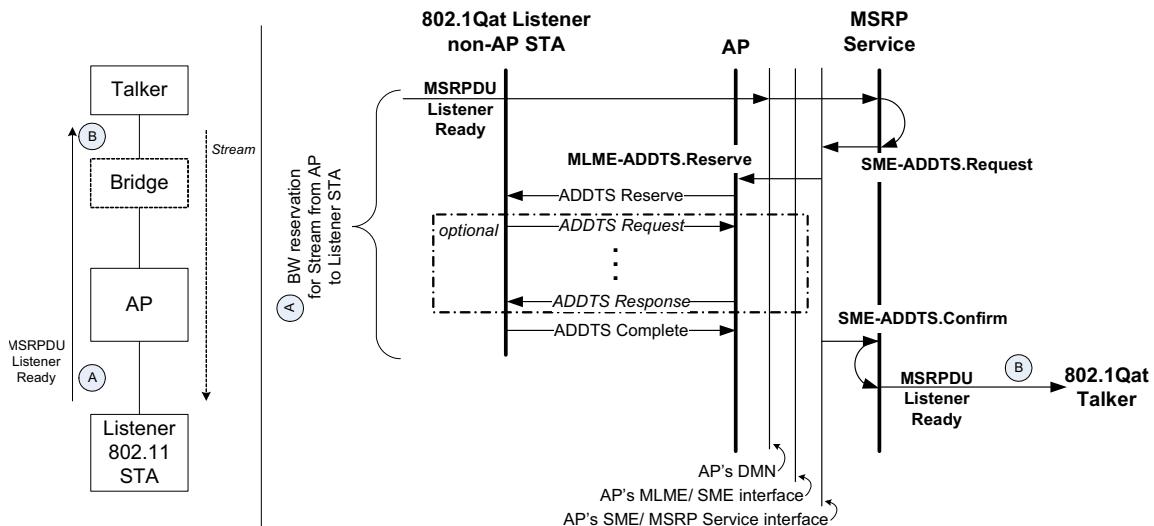


Figure C-13—MSRP/802.11 Listener STA to “Bridged” Talker Reservation Flows

The IEEE 802.11 message exchanges shown in the preceding three figures are explained in more detail as follows:

- a) BBS nodes identify MSRPDUs by their Group Destination Address (35.2.2.1) and EtherType (35.2.2.2) and send these PDUs to the AP.
- b) The AP forwards the MSRPDUs to the MSRP service on the DMN.
- c) The DMN translates the MSRP TSpec parameters into an equivalent IEEE 802.11 TSPEC and invokes DMN-SME interface primitives with the AP as follows:
 - 1) When the DMN receives a Talker Advertise message that originated from an upstream BSS node, the DMN invokes QoS Query transactions (SME-QUERY.Request) with the BSS QoS AP (i.e. HC where dot11RobustAvStreaming is set to true) to check whether or not the bandwidth advertised in the message's TSpec is available on each upstream to downstream node link of the BSS. In addition the DMN maps the MSRPDU's TSpec with the message's StreamID.
 - 2) When the DMN receives a Listener Ready message originated from a downstream BSS node, the DMN invokes a QoS Reservation transaction (SME-ADDTs.Request) with the BSS QoS manager to reserve the bandwidth associated with the message's StreamID on the downstream to upstream BSS node link.

The IEEE 802.11 BSS QoS AP on receipt of a SME-ADDTs.Request from the DMN shall make a determination about whether to accept the request or deny the request. The algorithm to be used by the BSS QoS AP to make this determination is an implementation detail.

If the BSS QoS AP decides to accept the request, the AP shall derive a medium time value from the parameters specified in the SME-ADDTs.Request. The BSS QoS AP shall then generate an autonomous ADDTS Reserve frame in which the medium time value is included and transmit it to the appropriate SRP Talker (BSS upstream) and Listener (BSS downstream) nodes.

If the BSS QoS AP decides to reject the request, it shall respond to the DMN with SME-ADDTs.confirm with a ResultCode of Rejected. The confirm primitive may also include a TSPEC which the BSS QoS AP can accept, if specified in a subsequent SME-ADDTs.request.

NOTE—The TSPEC included in the SME-ADDTs.confirm is based on the result of the negotiations (labeled optional in Figure C-11 through Figure C-13) with the upstream BSS node. As a result the TSPEC included in the SME-ADDTs.confirm may be different from the one in the SME-ADDTs.request from the DMN.

- d) After the DMN completes the BSS QoS transactions (SME-QUERY.Confirm or SME-ADDTs.Confirm as appropriate), the DMN behaves as an MSRP application on a Bridge and propagates MSRP attributes (35.2).

C.3.2 BSS DMN selection

The DMN shall be co-located with the device that supports the QoS AP function in the BSS.

C.3.3 BSS network bandwidth management

The MSRP service within the IEEE 802.11 network manages the BSS bandwidth for the MSRP streams by invoking the MLME QoS services. The DMN shall map the MLME services as described in Table C-4.

Table C-4—SRP to MLME QoS Services mapping

MSRP Attribute	MAD Primitive	SME QoS Services	Description
Talker Advertise	MAD_Join.request(new)	SME-QUERY	Query bandwidth without reservation
Listener Ready or Listener Ready Failed	MAD_Join.request(new)	SME-ADDS	Reserve bandwidth for a stream
Listener Ready or Listener Ready Failed	MAD_Join.request()	SME-ADDS ^a	Renew the bandwidth reservation (leased time) for a stream
Talker or Listener Leave	MAD_Leave.request()	SME-DELTS	Free bandwidth associated with a stream

^aBandwidth renewal is not required as long as the reservation is already established.

C.3.3.1 MSRPDU Encapsulation/De-encapsulation

In order to preserve the priority of an MSRPDU when traversing through a IEEE 802.11 network, the priority shall be encapsulated while the MSRPDU is in the IEEE 802.11 network and shall be de-encapsulated as it exits. See IEEE Std 802.11-2007 [B10] for additional information.

NOTE—For example if the User Priority of the MSRPDU is 4, CFI=0 and VLAN ID = 1893 the equivalent VLAN tag field (32 bits) is 81-00-87-65. When the frame enters the 802.11 network, the encapsulated 802.11 LLC header is AA-AA-03-00-00-81-00-87-65-AA-AA-03-00-00-00-08-00, where AA-AA-03-00-00-00-81-00-87-65 is the SNAP encoded VLAN header. When the frame exits the 802.11 network a de-encapsulation operation is performed and the resulting VLAN tag field is 81-00-87-65.

C.3.3.2 QoS Maintenance Report

An SRP DMN may obtain QoS Maintenance Report using IEEE 802.11 Transmit Stream/Category Measurement Requests and processing the corresponding Transmit Stream/Category Measurement Reports. The Transmit Stream/Category Measurement Request is sent to both the SRP Talker (BSS upstream) and the SRP Listener (BSS downstream) nodes. Triggers are set on appropriate conditions such that Transmit Stream/Category Measurement Reports are generated only when predefined thresholds are breached. See IEEE Std 802.11-2007 [B10], 7.3.2.21.10 Transmit Stream/Category Measurement Request, for details.

NOTE—This provides an indication to SRP (35.2.4) that bandwidth has changed.

C.3.3.3 SRP TSpec to IEEE 802.11 TSPEC mapping

SR Class B traffic has three parameters associated with it, namely delay budget per IEEE 802.11 hop (20msecs), **MaxFrameSize** (\leq 1500 bytes) and **MaxIntervalFrames** (units of 4000 frames per second). IEEE 802.11 TSPECs include a Minimum PHY rate that is derived from the SR Class B parameters as described below:

- a) Overhead = 10 byte VLAN tag + 8 byte Protocol definition = 18 bytes.
- b) Mean Data Rate = $(\text{SRP TSpec MaxFrameSize} + \text{overhead}) \times 4000 \times \text{SRP TSpec MaxIntervalFrames}$ bytes/sec.
- c) The Mean Data Rate is also the Max Data Rate (since it is assumed that MSDU size is fixed).
- d) Assuming 70% \times efficiency between the MAC and the PHY this translates into:
- e) $(10/7) \times (\text{SRP TSpec MaxFrameSize} + \text{overhead}) \times 4000 \times \text{SRP TSpec MaxIntervalFrames}$ bytes/sec, or

- f) $(10/7) \times 8 \times (\text{SRP TSpec MaxFrameSize} + \text{overhead}) \times 4000 \times \text{SRP TSpec MaxIntervalFrames}$ bits/sec.
- g) Minimum PHY Rate is therefore:
- h) $(10/7) \times 8 \times (\text{SRP TSpec MaxFrameSize} + \text{overhead}) \times 4000 \times \text{SRP TSpec MaxIntervalFrames}$ bits/sec

NOTE—For example, with 1500 and 1 for **MaxFrameSize** and **MaxIntervalFrames** the above turns into 69.394285 Mbps. Or, with 64 and 1 for **MaxFrameSize** and **MaxIntervalFrames** the above turns into 3.748571 Mbps

Table C-5 describes the mapping between SRP TSpec components and IEEE 802.11 QoS TSPEC parameters for the mandatory EDCA-AC mode.

Table C-5—EDCA-AC for AV Streams

TSPEC Parameter		SR Class B
TSINFO	TID	5
	Direction	Up, Down
	Access Policy	10 (EDCA)
	ACK Policy	10 (No Ack) / 11 (Block Ack)
	APSD	0
	Aggregation	Yes
	Priority	5
Nominal MSDU Size^a		SRP TSpec MaxFrameSize + 18 (also set bit 15 to a 1)
Maximum MSDU Size		SRP TSpec MaxFrameSize + 18
Peak Data Rate		(SRP TSpec MaxFrameSize + 18) × 4000 × SRP TSpec MaxIntervalFrames
Minimum PHY Rate^b		(10/7) × 8 × (SRP TSpec MaxFrameSize + 18) × 4000 × SRP TSpec MaxIntervalFrames
Delay Bound		20 msecs
Surplus Bandwidth Allowance^c		1.2

^abit15 set to indicate that the MSDU size is fixed

^bAssuming 70% efficiency between the MAC and the PHY

^c20% surplus allocation

Table C-6 describes the mapping between SRP TSpec components and IEEE 802.11 QoS TSPEC parameters for the optional HCCA mode.

Table C-6—HCCA for AV Streams

TSPEC Parameter		SR Class B
TSINFO	TID	5
	Direction	Up, Down
	Access Policy	01 (HCCA)
	ACK Policy	10 (No Ack) / 11 (Block Ack)
	APSD	0
	Aggregation	Yes
	Priority	5
Nominal MSDU Size ^a		0
Maximum MSDU Size		SRP TSpec MaxFrameSize + 18
Minimum Service Interval		10 msec
Maximum Service Interval		10 msec
Inactivity Interval		0
Minimum Data Rate		0
Mean Data Rate		0
Maximum Burst Size		$(\text{SRP TSpec MaxFrameSize} + 18) \times 4000 \times \text{SRP TSpec MaxIntervalFrames} \times 10^{-2}$
Minimum PHY Rate ^b		$(10/7) \times 8 \times (\text{SRP TSpec MaxFrameSize} + 18) \times 4000 \times \text{SRP TSpec MaxIntervalFrames}$
Peak Data Rate		$(\text{SRP TSpec MaxFrameSize} + 18) \times 4000 \times \text{SRP TSpec MaxIntervalFrames}$
Delay Bound		20 msec
Surplus Bandwidth Allowance ^c		1.2

^abit15 set to indicate that the MSDU size is fixed.^bAssuming 70% efficiency between the MAC and the PHY^c20% surplus allocation

Annex D

(normative)

IEEE 802.1 Organizationally Specific TLVs

D.1 Requirements of the IEEE 802.1 Organizationally Specific TLV sets

The IEEE 802.1 Organizationally Specific TLVs may be supported in conjunction with any of the destination MAC addresses identified in 7.1 of IEEE Std 802.1AB.

If any IEEE 802.1 Organizationally Specific TLV set is supported, all IEEE 802.1 Organizationally Specific TLVs that are identified as members of that TLV set shall be supported. All IEEE 802.1 Organizationally Specific TLVs shall conform to the LLDPDU bit and octet ordering conventions of 9.1 of IEEE Std 802.1AB.

The currently defined IEEE 802.1 Organizationally Specific TLVs are listed in Table D-1. The “TLV set name” column identifies the TLV set to which each TLV belongs. Any additions or changes to these TLVs will be included in this annex.

Table D-1— IEEE 802.1 Organizationally Specific TLVs

IEEE 802.1 subtype	TLV name	TLV set name	TLV reference	Feature clause reference
01	Port VLAN ID	basicSet	D.2.1	6.9
02	Port And Protocol VLAN ID	basicSet	D.2.2	6.12
03	VLAN Name	basicSet	D.2.3	12.10.2.1.3
04	Protocol Identity	basicSet	D.2.4	D.2.4
05	VID Usage Digest	basicSet	D.2.5	D.2.5
06	Management VID	basicSet	D.2.6	D.2.6
07	Link Aggregation	basicSet	D.2.7	D.2.7, IEEE Std 802.1AX-2008
08	Congestion Notification	cnSet	D.2.8	Clause 33
09–FF	Reserved	—	—	

D.2 Organizationally Specific TLV definitions

D.2.1 Port VLAN ID TLV

The Port VLAN ID TLV is an optional fixed length TLV that allows a VLAN bridge port to advertise the port’s VLAN identifier (PVID) that is associated with untagged or priority tagged frames (see 6.9).

Figure D-1 shows the Port VLAN ID TLV format.

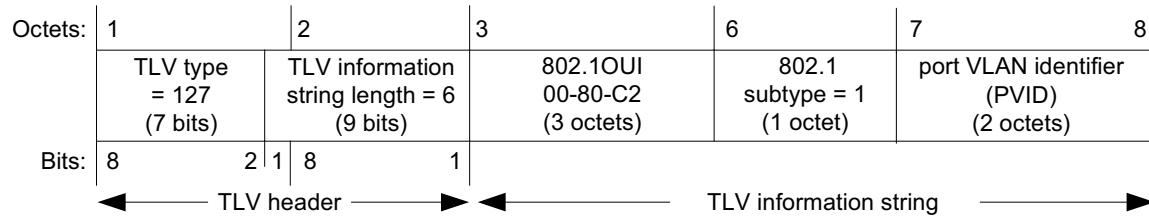


Figure D-1—Port VLAN ID TLV Format

D.2.1.1 port VLAN identifier (PVID)

The port VLAN identifier field shall contain the VLAN ID for the bridge port as defined in 6.9. A value of zero shall be used if the system either does not know the PVID or does not support port-based VLAN operation.

D.2.1.2 Port VLAN ID usage rules

An LLDPDU should contain no more than one Port VLAN ID TLV.

D.2.2 Port And Protocol VLAN ID TLV

The Port And Protocol VLAN ID TLV is an optional TLV that allows a bridge port to advertise a port and protocol VLAN ID. Figure D-2 shows the Port And Protocol VLAN ID TLV format.

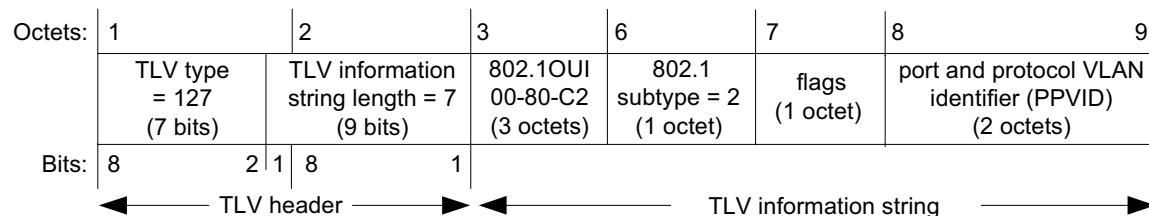


Figure D-2—Port And Protocol VLAN ID TLV Format

D.2.2.1 flags

The flags field shall contain a bit map indicating the port and protocol VLAN capability and status as defined in Table D-2.

Table D-2—Port and protocol capability/status

Bit	Function	Value/meaning
1	Port and protocol VLAN supported	1 = supported 0 = not supported
2	Port and protocol VLAN enabled	1 = enabled 0 = not enabled
3–8	reserved for future standardization	(set to zero)

D.2.2.2 port and protocol VLAN ID (PPVID)

The port and protocol VLAN ID field shall contain the PPVID number for this IEEE 802 LAN station. If the port is not capable of supporting port and protocol VLANs and/or the port is not enabled with any port and protocol VLAN, the PPVID number should be zero.

D.2.2.3 Port And Protocol VLAN ID TLV usage rules

Port And Protocol VLAN ID TLVs are subject to the following rules:

- a) If more than one Port And Protocol VLAN ID TLV is defined for a port, the PPVID value shall be different from any other PPVID defined for the port.
- b) If the support bit (bit 1) of the flag field indicates that the port is not capable of supporting port and protocol VLANs but the enabled bit (bit 2) indicates that the port is enabled with one or more port and protocol VLANs, the Port And Protocol VLAN ID TLV is interpreted as containing an error and will be discarded.
- c) If the PPVID reference number is greater than 4094, the Port And Protocol VLAN ID TLV is interpreted as containing an error and is discarded.

D.2.3 VLAN Name TLV

The VLAN Name TLV is an optional TLV that allows an IEEE 802.1Q-compatible IEEE 802 LAN station to advertise the assigned name of any VLAN with which it is configured.

Figure D-3 shows the VLAN Name TLV format.

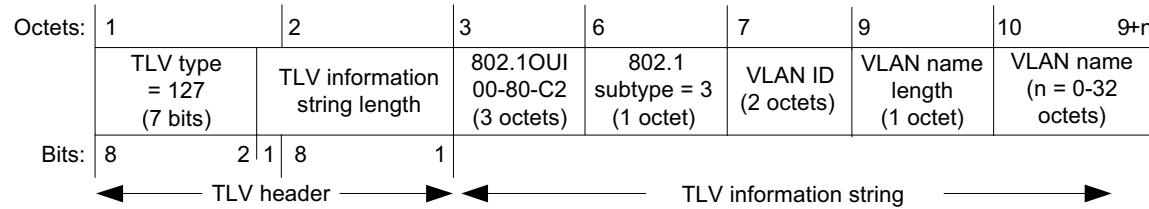


Figure D-3—VLAN Name TLV format

D.2.3.1 TLV information string length

The TLV information string length field shall contain the length, in octets, of the (VLAN name + 7).

D.2.3.2 VLAN ID (VID)

The VLAN ID field shall contain the VID number associated with the VLAN name.

D.2.3.3 VLAN name length

The VLAN name length field shall contain the length, in octets, of the VLAN name. A VLAN name length of zero indicates that there is no VLAN name associated with this VLAN.

D.2.3.4 VLAN name

If present, the VLAN name field shall contain the VLAN's name. If implementations support IETF RFC 4363, the dot1QVLANStaticName object should be used for this field.

D.2.3.5 VLAN Name TLV usage rules

If more than one VLAN Name TLV is defined for a port, the VLAN ID and the associated VLAN name combination shall be different from any other VLAN ID and VLAN name combination defined for the port.

D.2.4 Protocol Identity TLV

The Protocol Identity TLV is an optional TLV that allows an IEEE 802 LAN station to advertise particular protocols that are accessible through the port. Figure D-4 shows the protocol identity TLV format.

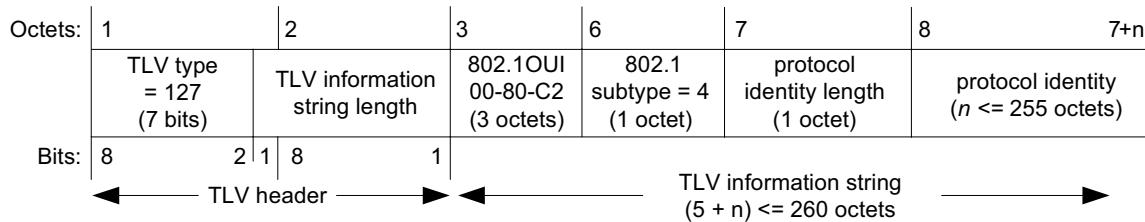


Figure D-4—Protocol Identity TLV format

D.2.4.1 TLV information string length

The TLV information string length field shall contain the length, in octets, of the (protocol identity + 5)

D.2.4.2 protocol identity length

The protocol identity length field shall contain the length, in octets, of the protocol identity.

D.2.4.3 protocol identity

The protocol identity field shall contain the first n octets of the protocol after the layer 2 addresses (i.e. for example, starting with the EtherType field) that the sender would like to advertise. The value of n is determined by the need for the protocol to disambiguate itself. The protocol information string shall include enough octets to allow the receiver to correctly identify the protocol and its version. To advertise Spanning Tree Protocols, for example, the protocol identity field would need to include at least eight octets: IEEE 802.3 length (two octets), LLC addresses (two octets), IEEE 802.3 control (one octet), Protocol ID (two octets), and the protocol version (one octet).

D.2.4.4 Protocol Identity TLV usage rules

If more than one Protocol Identity TLV is defined for a port, the protocol identity field value shall be different from any other Protocol Identity TLV defined for the port.

D.2.5 VID Usage Digest TLV

The VID Usage Digest TLV is an optional TLV that allows an IEEE Std 802.1Q-compatible IEEE 802 LAN station to advertise the value of a VID Usage Digest associated with the system. The value of the VID Usage Digest is obtained by applying the CRC32 function (IEEE Std 802.3-2008, 4.2.10) to a VID Usage Table having a fixed length of 128 octets. A bit of the VID Usage Table contains the value PBB-TE-USAGE (binary 1) if the corresponding element of the MST Configuration Table (8.9.1) contains the value PBB-TE MSTID (hex FFE) and otherwise contains the value NON-PBB-TE-USAGE (binary 0).

Figure D-5 shows the VID Usage Digest TLV format.

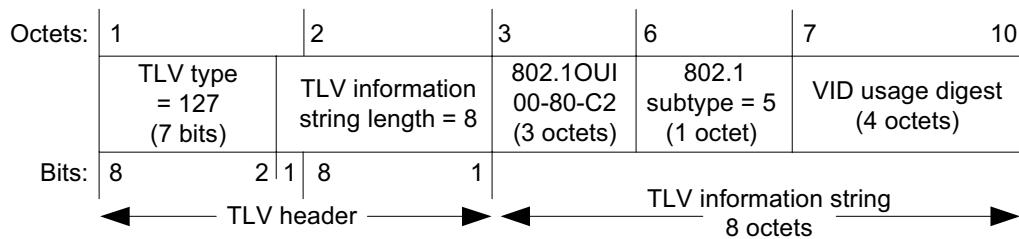


Figure D-5—VID Usage Digest TLV format

D.2.5.1 VID Usage Digest

The VID Usage Digest field shall contain a VID Usage Digest value obtained by applying the CRC32 function to the 128-octet VID Usage Table. A bit of the VID Usage Table contains the value PBB-TE-USAGE (binary 1) if the corresponding element of the MST Configuration Table (8.9.1) contains the value PBB-TE MSTID (hex FFE) and otherwise contains the value NON-PBB-TE-USAGE (binary 0).

D.2.6 Management VID TLV

The Management VID TLV is an optional TLV that allows an IEEE 802.1Q-compatible IEEE 802 LAN station to advertise the value of a Management VID associated with the system.

Figure D-6 shows the Management VID TLV format.

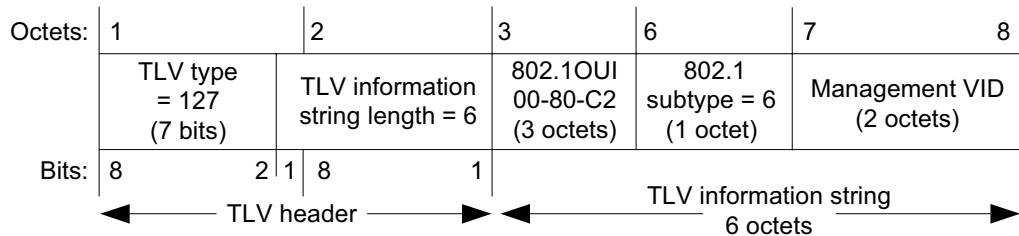


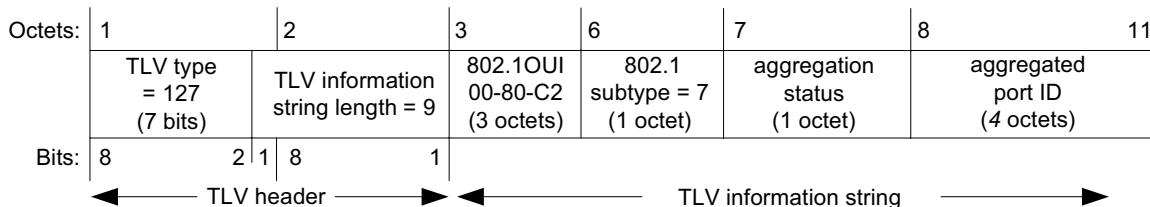
Figure D-6—Management VID TLV format

D.2.6.1 Management VID

The Management VID field shall contain the value configured for the Management VID, or the value 0 if a Management VID has not been provisioned.

D.2.7 Link Aggregation TLV

The Link Aggregation TLV indicates whether the link is capable of being aggregated, whether the link is currently in an aggregation, as specified in IEEE Std 802.1AX, and if in an aggregation, the port identification of the aggregation. Figure D-7 shows the format for this TLV.

**Figure D-7—Link Aggregation TLV format****D.2.7.1 aggregation status**

The link aggregation status field shall contain a bit map of the link aggregation capabilities and the current aggregation status of the link as defined in Table D-3.

Table D-3—Link aggregation capability/status

Bit	Function	Value/meaning
0	Aggregation capability	0 = not capable of being aggregated 1 = capable of being aggregated
1	Aggregation status	0 = not currently in aggregation 1 = currently in aggregation
2–7	reserved for future standardization	—

D.2.7.2 aggregated port ID

The aggregated port ID field shall contain the IEEE 802.3 aggregated port identifier, aAggPortID, derived from the ifNumber in the ifIndex for the interface.

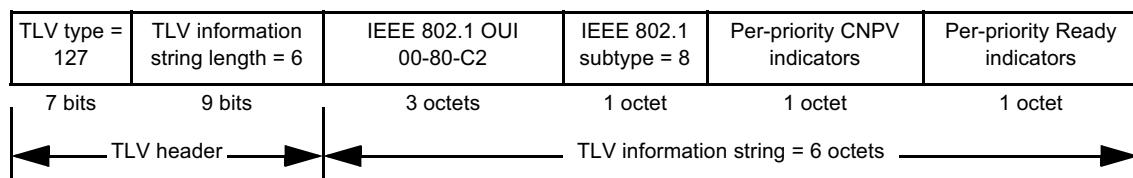
NOTE—This field is meaningful only if the link concerned is currently a member of an aggregation.

D.2.7.3 Link Aggregation TLV usage rules

An LLDPDU should contain no more than one Link Aggregation TLV.

D.2.8 Congestion Notification TLV

The TLV illustrated in Figure D-8 is encoded into each IEEE Std 802.1AB LLDP message transmitted by a system that is configured to support congestion notification on one or more priority values.

**Figure D-8—Congestion Notification TLV format**

A system shall transmit no more than one Congestion Notification TLV in a single LLDP PDU. If a system receives an LLDP PDU with more than one Congestion Notification TLV, the behavior of the receiving system is unspecified. A system shall not transmit a Congestion Notification TLV with all of the Per-priority CNPV indicators (D.2.8.3) clear (0).

D.2.8.1 TLV type

A 7-bit integer value occupying the most significant bits of the first octet of the TLV. Always contains the value 127.

D.2.8.2 TLV information string length

A 9-bit unsigned integer, occupying the LSB of the first octet of the TLV (the MSB of the TLV information string length) and the entire second octet of the TLV, containing the total number of octets in the TLV information string of the Congestion Notification TLV. This does not count the TLV type and TLV information string length fields. It is equal to 6.

D.2.8.3 Per-priority CNPV indicators

A bit vector, one per priority value, containing all eight of the cnpdXmitCnpvCapable variables (32.4.7) for this port. The LSB of the octet carries priority 0's cnpdXmitCnpvCapable variable, and the MSB that of priority 7.

D.2.8.4 Per-priority Ready indicators

A bit vector, one per priority value, containing all eight of the cnpdXmitReady variables (32.4.8) for this port. The LSB of the octet carries priority 0's cnpdXmitReady variable, and the MSB that of priority 7.

D.3 IEEE 802.1 Organizationally Specific TLV management

D.3.1 IEEE 802.1 Organizationally Specific TLV selection management

TLV selection management consists of providing the network manager with the means to select which specific IEEE 802.1 Organizationally Specific TLVs are enabled for inclusion in an LLDPDU. The following LLDP variables cross reference to LLDP local systems configuration MIB tables indicate which specific TLVs are enabled for the particular port(s) on the system. The specific port(s) through which each TLV is enabled for transmission may be set (or reset) by the network manager:

- a) **mibXdot1PortVlanTxEnable:** This variable lists the VLAN ID of the port through which the referenced TLV is enabled for transmission.
- b) **mibXdot1VlanNameConfigTxEnable:** This variable lists the different VLAN name/PPVID TLVs that are defined for the system, each with a bit map indicating the system ports through which the particular VLAN name TLV is enabled for transmission.
- c) **mibXdot1ProtoVlanConfigTxEnable:** This variable lists the port and protocol VLAN TLVs that are defined for the system, each with a bit map indicating the system ports through which the particular port and protocol VLAN TLV is enabled for transmission.
- d) **mibXdot1ProtocolConfigTxEnable:** This variable lists the protocol identity TLVs that are defined for the system, each with a bit map indicating the system ports through which the particular protocol TLV is enabled for transmission.

D.3.2 IEEE 802.1 managed objects—TLV variables

D.3.2.1 Port VLAN ID TLV managed objects

- a) **PVID:** The port VLAN identifier (see D.2.1.1).

D.3.2.2 Port And Protocol VLAN ID TLV managed objects

- a) **Port and protocol VLAN supported:** A flag indicating whether port and protocol VLANs are supported (see D.2.2.1).
- b) **Port and protocol VLAN enabled:** A flag indicating whether port and protocol VLANs are enabled (see D.2.2.1).
- c) **PPVID:** The advertised port and protocol VLAN ID (see D.2.2.2).

D.3.2.3 VLAN Name TLV managed objects

- a) **VID:** The VLAN ID associated with the VLAN name (see D.2.3.2).
- b) **VLAN name length:** The length of the VLAN name (see D.2.3.3).
- c) **VLAN name:** The VLAN's name (see D.2.3.4).

D.3.2.4 Protocol Identity TLV managed objects

- a) **protocol identity length:** The length of the protocol identity (see D.2.4.2).
- b) **protocol identity:** The protocol's identity (see D.2.4.3).

D.3.2.5 VID Usage Digest TLV managed objects

- a) VID Usage Digest: The VID Usage Digest value (see D.2.5.1)

D.3.2.6 Management VID TLV managed objects

- a) Management VID: The Management VID value (see D.2.6.1)

D.3.2.7 Link Aggregation TLV managed objects

- a) **aggregation status:** The capability and current aggregation status of the link (see D.2.7.1).
- b) **aggregated port ID:** The aggregated port identifier (see D.2.7.2).

D.3.2.8 Congestion Notification TLV managed objects

- a) **CNPV indicators:** A per-priority CNPV indicator vector (see D.2.8.3).
- b) **CN ready indicators:** A per-priority ready indicator vector (see D.2.8.4).

D.4 IEEE 802.1/LLDP extension MIB

D.4.1 Internet Standard Management Framework

LLDP MIBs are designed to operate in a manner consistent with the principles of the Internet Standard Management Framework, which describes the separation of a data modeling language (for example, SMIv2) from content specific data models (for example the LLDP remote systems MIB), and from messages and protocol operations used to manipulate the data (for example SNMPv3).

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).

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 IETF STD 58, IETF RFC 2578 (1999), IETF RFC 2579 (1999), and IETF RFC 2580 (1999).

D.4.2 Structure of the IEEE 802.1/LLDP extension MIB

Table D-4 summarizes the particular object groups that are required for each operating mode. The implemented MIB shall comply with the MIB conformance section for the particular operating mode being supported.

Table D-4—IEEE 802.1 extension MIB object group conformance requirements

MIB group	Rx mode	Tx mode	Tx/Rx mode
lldpV2Xdot1ConfigGroup	M ^a	M	M
lldpV2Xdot1LocSysGroup	M	—	M
lldpV2Xdot1RemSysGroup	—	M	M
ifGeneralInformationGroup	M	M	M

^aM=Mandatory

Table D-5 shows the structure of the MIB and the relationship of the MIB objects to the LLDP operational status/control variables, LLDP statistics variables, and TLV variables.

Table D-5—IEEE 802.1/LLDP extension MIB object cross reference

MIB table	MIB object	LLDP reference
<i>Configuration group</i>		
lldpV2Xdot1ConfigPortVlanTable		Augments lldpV2Xdot1ConfigPortVlanTable
	lldpV2Xdot1ConfigPortVlanTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB
lldpV2Xdot1ConfigVlanNameTable		Augments lldpV2Xdot1LocVlanNameEntry
	lldpV2Xdot1ConfigVlanNameTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB
lldpV2Xdot1ConfigProtoVlanTable		Augments lldpV2Xdot1LocProtoVlanEntry
	lldpV2Xdot1ConfigProtoVlanTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB
lldpV2Xdot1ConfigProtocolTable		Augments lldpV2Xdot1LocProtocolEntry
	lldpV2Xdot1ConfigProtocolTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB
lldpV2Xdot1ConfigVidUsageDigestTable		Augments lldpV2Xdot1LocVidUsageDigestEntry
	lldpV2Xdot1ConfigVidUsageDigestTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB

Table D-5—IEEE 802.1/LDP extension MIB object cross reference (continued)

MIB table	MIB object	LLDP reference
lldpV2Xdot1ConfigManVidTable		Augments lldpV2Xdot1LocManVidEntry
	lldpV2Xdot1ConfigManVidTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB
lldpXdot1CnConfigCnTable		Augments lldpV2Xdot1LocManVidEntry
	lldpXdot1CnConfigCnTxEnable	Normal LLPDUs, 9.1.2.1 of IEEE Std 802.1AB
<i>Local system information</i>		
lldpV2Xdot1LocTable		D.2.1
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocPortVlanId	port VLAN identifier, D.2.1.1
lldpV2Xdot1LocProtoVlanTable		D.2.2
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocProtoVlanId	port and protocol VLAN ID, D.2.2.2
	lldpV2Xdot1LocProtoVlanSupported	flags, D.2.2.1
	lldpV2Xdot1LocProtoVlanEnabled	flags, D.2.2.1
lldpV2Xdot1LocVlanNameTable		D.2.3
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocVlanId	VLAN ID, D.2.3.2 (Table index)
	lldpV2Xdot1LocVlanName	VLAN name, D.2.3.4
lldpV2Xdot1LocProtocolTable		D.2.4
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocProtocolIndex	(Table index)
	lldpV2Xdot1LocProtocolId	protocol identity, D.2.4.3
lldpV2Xdot1LocVidUsageDigestTable		D.2.5
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocVidUsageDigest	VID usage digest, D.2.5.1
lldpV2Xdot1LocManVidTable		D.2.6
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocManVid	Management VID, D.2.6.1
lldpV2Xdot1LocLinkAggTable		D.2.7
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocLinkAggStatus	aggregation status, D.2.7.1
	lldpV2Xdot1LocLinkAggPortId	aggregated port ID, D.2.7.2

Table D-5—IEEE 802.1/LDP extension MIB object cross reference (continued)

MIB table	MIB object	LLDP reference
lldpV2Xdot1LocCnTable		D.2.8
<i>Remote system information</i>	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1LocCNPVIndicators	CPNV indicators, D.2.8.3
	lldpV2Xdot1LocReadyIndicators	Ready indicators, D.2.8.4
lldpV2Xdot1RemTable		D.2.1
<i>Remote system information</i>	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2RemIndex	(Table index)
	lldpV2Xdot1RemPortVlanId	port VLAN identifier, D.2.1.1
lldpV2Xdot1RemProtoVlanTable		D.2.2
<i>Remote system information</i>	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2RemIndex	(Table index)
	lldpV2Xdot1RemProtoVlanId	port and protocol VLAN ID, D.2.2.2 (Table index)
	lldpV2Xdot1RemProtoVlanSupported	flags, D.2.2.1
	lldpV2Xdot1RemProtoVlanEnabled	flags, D.2.2.1
lldpV2Xdot1RemVlanNameTable		D.2.3
<i>Remote system information</i>	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2RemIndex	(Table index)
	lldpV2Xdot1RemVlanId	VLAN ID, D.2.3.2 (Table index)
	lldpV2Xdot1RemVlanName	VLAN name, D.2.3.4
lldpV2Xdot1RemProtocolTable		D.2.4
<i>Remote system information</i>	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2RemIndex	(Table index)
	lldpV2Xdot1RemProtocolIndex	(Table index)
	lldpV2Xdot1RemProtocolId	protocol identity, D.2.4.3

Table D-5—IEEE 802.1/LDP extension MIB object cross reference (continued)

MIB table	MIB object	LLDP reference
lldpV2Xdot1RemVidUsageDigestTable		D.2.5
lldpV2Xdot1RemManVidTable	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2Xdot1RemVidUsageDigest	VID usage digest, D.2.5.1
lldpV2Xdot1RemLinkAggTable		D.2.6
lldpV2Xdot1RemCnTable	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2RemIndex	(Table index)
lldpV2Xdot1RemLinkAggStatus		aggregation status, D.2.7.1
lldpV2Xdot1RemLinkAggPortId		aggregation port ID, D.2.7.2
lldpV2Xdot1RemCnTable		D.2.8
lldpV2Xdot1RemCnTable	lldpV2RemTimeMark	(Table index)
	lldpV2RemLocalIfIndex	(Table index)
	lldpV2RemLocalDestMACAddress	(Table index)
	lldpV2LocPortIfIndex	(Table index)
	lldpV2Xdot1RemCNPVIndicators	CPNV indicators, D.2.8.3
	lldpV2Xdot1RemReadyIndicators	Ready indicators, D.2.8.4

D.4.3 Relationship to other MIBs

Version 2 of the IEEE 802.1 LLDP extension MIB module appears in D.4.5. Version 1 of the MIB module that was published in the initial 2005 publication of IEEE Std 802.1AB has been superseded by version 2 of the MIB module, and support of the version 2 module is a requirement for conformance to the required or optional capabilities (Clause 5 of IEEE Std 802.1AB) in the 2009 revision of IEEE Std 802.1AB. The version 2 MIB module reflects changes in indexation of the MIB objects that support the use of LLDP with multiple destination MAC addresses, as discussed in Clause 6 of IEEE Std 802.1AB.

The relationship of the IEEE 802.1 LLDP extension MIB module to other MIBs, and coexistence between version 1 and version 2 implementations, is further discussed in 11.3 of IEEE Std 802.1AB.

D.4.4 Security considerations for IEEE 802.1 LLDP extension MIB module

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write⁴⁸. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a nonsecure environment without proper protection can have a negative effect on network operations.

Setting the following objects to incorrect values can result in improper operation of LLDP when in the transmit mode:

- a) lldpV2Xdot1ConfigPortVlanTxEnableV2
- b) lldpV2Xdot1ConfigVlanNameTxEnable
- c) lldpV2Xdot1ConfigProtoVlanTxEnable
- d) lldpV2Xdot1ConfigProtocolTxEnable
- e) lldpV2Xdot1ConfigVidUsageDigestTxEnable
- f) lldpV2Xdot1ConfigManVidTxEnable

The following readable objects in this MIB module may be considered to be sensitive or vulnerable in some network environments:

- g) MIB objects that are related to the transmit mode
 - 1) lldpV2Xdot1LocPortVlanId
 - 2) lldpV2Xdot1LocProtoVlanSupported
 - 3) lldpV2Xdot1LocProtoVlanEnabled
 - 4) lldpV2Xdot1LocVlanName
 - 5) lldpV2Xdot1LocProtocolId
 - 6) lldpV2Xdot1LocVidUsageDigest
 - 7) lldpV2Xdot1LocManVidTxEnable
 - 8) lldpV2Xdot1LocLinkAggStatus
 - 9) lldpV2Xdot1LocLinkAggPortId
- h) MIB objects that are related to the receive mode
 - 1) lldpV2Xdot1RemPortVlanId
 - 2) lldpV2Xdot1RemProtoVlanSupported
 - 3) lldpV2Xdot1RemProtoVlanEnabled
 - 4) lldpV2Xdot1RemVlanName
 - 5) lldpV2Xdot1RemProtocolId
 - 6) lldpV2Xdot1RemVidUsageDigest
 - 7) lldpV2Xdot1RemManVidTxEnable
 - 8) lldpV2Xdot1RemLinkAggStatus
 - 9) lldpV2Xdot1RemLinkAggPortId

This concern applies both to objects that describe the configuration of the local host, as well as for objects that describe information from the remote hosts, acquired via LLDP and displayed by the objects in this MIB module. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPSec), even then, 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 this MIB module.

⁴⁸In IETF MIB definitions, the MAX-ACCESS clause defines the type of access that is allowed for particular data elements in the MIB. An explanation of the MAX-ACCESS mappings is given in section 7.3 of IETF RFC 2578 (1999).

Implementers should consider the security features as provided by the SNMPv3 framework (see IETF RFC 3410 (2002), section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, implementers should not deploy SNMP versions prior to SNMPv3. Instead, implementers should deploy SNMPv3 to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

D.4.5 IEEE 802.1 LLDP extension MIB module—version 2^{49,50}

In the following MIB definition, should any discrepancy between the DESCRIPTION text and the corresponding definition in D.2.1 through D.4 occur, the definition in D.2.1 through D.4 shall take precedence.

```
LLDP-EXT-DOT1-V2-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32
        FROM SNMPv2-SMI
    TruthValue,
    TEXTUAL-CONVENTION
        FROM SNMPv2-TC
    SnmpAdminString
        FROM SNMP-FRAMEWORK-MIB
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    ifGeneralInformationGroup
        FROM IF-MIB
    lldpV2Extensions,
    lldpV2LocPortIfIndex,
    lldpV2RemTimeMark,
    lldpV2RemLocalIfIndex,
    lldpV2RemLocalDestMACAddress,
    lldpV2RemIndex,
    lldpV2PortConfigEntry
        FROM LLDP-V2-MIB
    VlanId
        FROM Q-BRIDGE-MIB
    LldpV2LinkAggStatusMap
        FROM LLDP-V2-TC-MIB;

lldpV2Xdot1MIB MODULE-IDENTITY
LAST-UPDATED "201103230000Z" -- March 23, 2011
ORGANIZATION "IEEE 802.1 Working Group"
CONTACT-INFO
    "WG-URL: http://grouper.ieee.org/groups/802/1/index.html
    WG-Email: STDS-802-1-L@LISTSERV.IEEE.ORG
```

⁴⁹*Copyright release for MIBs:* Users of this standard may freely reproduce the MIB contained in this subclause so that it can be used for its intended purpose.

⁵⁰An ASCII version of this MIB module can be obtained by Web browser from the IEEE 802.1 Website at <http://www.ieee802.org/1/pages/MIBS.html>.

Contact: Tony Jeffree
Postal: C/O IEEE 802.1 Working Group
IEEE Standards Association
445 Hoes Lane
P.O. Box 1331
Piscataway
NJ 08855-1331
USA

E-mail: STDS-802-1-L@LISTSERV.IEEE.ORG"

DESCRIPTION

"The LLDP Management Information Base extension module for IEEE 802.1 organizationally defined discovery information.

In order to assure the uniqueness of the LLDP-V2-MIB, llldpV2Xdot1MIB is branched from llldpV2Extensions using an Organizationally Unique Identifier (OUI) value as the node. An OUI is a 24 bit globally unique number assigned by the IEEE Registration Authority - see:

<http://standards.ieee.org/develop/regauth/oui/index.html>

Unless otherwise indicated, the references in this MIB module are to IEEE Std 802.1Q-2011.

Copyright (C) IEEE (2011). This version of this MIB module is published as Annex D.4.5 of IEEE Std 802.1Q-2011; see the standard itself for full legal notices."

REVISION "201103230000Z" -- March 23, 2011

DESCRIPTION

"Published as part of IEEE Std 802.1Q-2011 revision. This revision contains changes associated with relocating the extension MIB from IEEE Std 802.1AB to IEEE Std 802.1Q, minor tweaks to the text of the DESCRIPTION statement above to fix references to IEEE Std 802.1Q, updating of references to refer to Annex D, and addition of object definitions for Congestion Notification TLVs and corresponding compliance statements."

REVISION "200906080000Z" -- June 08, 2009

DESCRIPTION

"Published as part of IEEE Std 802.1AB-2009 revision. This revision incorporated changes to the MIB to support the use of LLDP with multiple destination MAC addresses, and to import the Link Aggregation TLV from the 802.3 extension MIB"

-- OUI for IEEE 802.1 is 32962 (00-80-C2)
::= { llldpV2Extensions 32962 }

--

-- Organizationally Defined Information Extension - IEEE 802.1
-- Definitions to support the basicSet TLV set (Table D-1)

```

--  

--  

--  

1lldpV2Xdot1Objects      OBJECT IDENTIFIER ::= { lldpV2Xdot1MIB 1 }

-- LLDP IEEE 802.1 extension MIB groups
1lldpV2Xdot1Config      OBJECT IDENTIFIER ::= { lldpV2Xdot1Objects 1 }
1lldpV2Xdot1LocalData   OBJECT IDENTIFIER ::= { lldpV2Xdot1Objects 2 }
1lldpV2Xdot1RemoteData  OBJECT IDENTIFIER ::= { lldpV2Xdot1Objects 3 }

--  

-- IEEE 802.1 - Configuration for the basicSet TLV set
--  

--  

--  

-- 1lldpV2Xdot1ConfigPortVlanTable : configure the transmission of the
--                                  Port VLAN-ID TLVs on set of ports.
--  

--  

1lldpV2Xdot1ConfigPortVlanTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF L1dpV2Xdot1ConfigPortVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that controls selection of LLDP Port VLAN-ID TLVs
         to be transmitted on individual ports."
    ::= { lldpV2Xdot1Config 1 }

1lldpV2Xdot1ConfigPortVlanEntry OBJECT-TYPE
    SYNTAX      L1dpV2Xdot1ConfigPortVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "LLDP configuration information that controls the
         transmission of IEEE 802.1 organizationally defined Port
         VLAN-ID TLV on LLDP transmission capable ports.

This configuration object augments the lldpV2PortConfigEntry of
the LLDP-MIB, therefore it is only present along with the port
configuration defined by the associated lldpV2PortConfigEntry
entry.

Each active lldpConfigEntry is restored from non-volatile
storage (along with the corresponding lldpV2PortConfigEntry)
after a re-initialization of the management system."
AUGMENTS { lldpV2PortConfigEntry }
    ::= { lldpV2Xdot1ConfigPortVlanTable 1 }

L1dpV2Xdot1ConfigPortVlanEntry ::= SEQUENCE {
    1lldpV2Xdot1ConfigPortVlanTxEnable  TruthValue
}

1lldpV2Xdot1ConfigPortVlanTxEnable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The lldpV2Xdot1ConfigPortVlanTxEnable, which is defined

```

as a truth value and configured by the network management, determines whether the IEEE 802.1 organizationally defined port VLAN TLV transmission is allowed on a given LLDP transmission capable port.

The value of this object is restored from non-volatile storage after a re-initialization of the management system."

REFERENCE

"9.1.2.1 of IEEE Std 802.1AB"

DEFVAL { false }

::= { lldpV2Xdot1ConfigPortVlanEntry 1 }

--
-- lldpV2Xdot1ConfigVlanNameTable : configure the transmission of the
-- VLAN name instances on set of ports.
--

lldpV2Xdot1ConfigVlanNameTable OBJECT-TYPE
SYNTAX SEQUENCE OF LldpV2Xdot1ConfigVlanNameEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The table that controls selection of LLDP VLAN name TLV
instances to be transmitted on individual ports."
 ::= { lldpV2Xdot1Config 2 }

lldpV2Xdot1ConfigVlanNameEntry OBJECT-TYPE
SYNTAX LldpV2Xdot1ConfigVlanNameEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"LLDP configuration information that specifies the set of
ports (represented as a PortList) on which the Local System
VLAN name instance is transmitted.

This configuration object augments the lldpV2LocVlanEntry, therefore it is only present along with the VLAN Name instance contained in the associated lldpV2LocVlanNameEntry entry.

Each active lldpV2Xdot1ConfigVlanNameEntry is restored from non-volatile storage (along with the corresponding lldpV2Xdot1LocVlanNameEntry) after a re-initialization of the management system."

AUGMENTS { lldpV2Xdot1LocVlanNameEntry }
 ::= { lldpV2Xdot1ConfigVlanNameTable 1 }

LldpV2Xdot1ConfigVlanNameEntry ::= SEQUENCE {
 lldpV2Xdot1ConfigVlanNameTxEnable TruthValue
}

lldpV2Xdot1ConfigVlanNameTxEnable OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The boolean value that indicates whether the corresponding

Local System VLAN name instance is transmitted on the port defined by the given lldpV2Xdot1LocVlanNameEntry.

The value of this object is restored from non-volatile storage after a re-initialization of the management system."

REFERENCE

"9.1.2.1 of IEEE Std 802.1AB"

DEFVAL { false }

::= { lldpV2Xdot1ConfigVlanNameEntry 1 }

```
--  
-- lldpV2Xdot1ConfigProtoVlanTable : configure the transmission of the  
-- protocol VLAN instances on set  
-- of ports.  
--
```

lldpV2Xdot1ConfigProtoVlanTable OBJECT-TYPE
SYNTAX SEQUENCE OF LldpV2Xdot1ConfigProtoVlanEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The table that controls selection of LLDP Port and Protocol
VLAN ID TLV instances to be transmitted on individual ports."
::= { lldpV2Xdot1Config 3 }

lldpV2Xdot1ConfigProtoVlanEntry OBJECT-TYPE
SYNTAX LldpV2Xdot1ConfigProtoVlanEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"LLDP configuration information that specifies the set of
ports (represented as a PortList) on which the Local System
Protocol VLAN instance is transmitted.

This configuration object augments the lldpV2Xdot1LocVlanEntry, therefore it is only present along with the Port and Protocol VLAN ID instance contained in the associated lldpV2Xdot1LocVlanEntry entry.

Each active lldpV2Xdot1ConfigProtoVlanEntry is restored from non-volatile storage (along with the corresponding lldpV2Xdot1LocProtoVlanEntry) after a re-initialization of the management system."

AUGMENTS { lldpV2Xdot1LocProtoVlanEntry }
::= { lldpV2Xdot1ConfigProtoVlanTable 1 }

LldpV2Xdot1ConfigProtoVlanEntry ::= SEQUENCE {
 lldpV2Xdot1ConfigProtoVlanTxEnable TruthValue
}

lldpV2Xdot1ConfigProtoVlanTxEnable OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-write
STATUS current
DESCRIPTION
"The boolean value that indicates whether the corresponding

Local System Port and Protocol VLAN instance is transmitted on the port defined by the given 1ldpV2Xdot1LocProtoVlanEntry.

The value of this object is restored from non-volatile storage after a re-initialization of the management system."

REFERENCE

"9.1.2.1 of IEEE Std 802.1AB"

DEFVAL { false }
 ::= { 1ldpV2Xdot1ConfigProtoVlanEntry 1 }

--
-- 1ldpV2Xdot1ConfigProtocolTable : configure the transmission of the
-- protocol instances on set
-- of ports.
--

1ldpV2Xdot1ConfigProtocolTable OBJECT-TYPE
 SYNTAX SEQUENCE OF L1dpV2Xdot1ConfigProtocolEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "The table that controls selection of LLDP Protocol
 TLV instances to be transmitted on individual ports."
 ::= { 1ldpV2Xdot1Config 4 }

1ldpV2Xdot1ConfigProtocolEntry OBJECT-TYPE
 SYNTAX L1dpV2Xdot1ConfigProtocolEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "LLDP configuration information that specifies the set of
 ports (represented as a PortList) on which the Local System
 Protocol instance is transmitted.

This configuration object augments the 1ldpV2Xdot1LocProtoEntry, therefore it is only present along with the Protocol instance contained in the associated 1ldpV2Xdot1LocProtoEntry entry.

Each active 1ldpV2Xdot1ConfigProtocolEntry is restored from non-volatile storage (along with the corresponding 1ldpV2Xdot1LocProtocolEntry) after a re-initialization of the management system."
AUGMENTS { 1ldpV2Xdot1LocProtocolEntry }
 ::= { 1ldpV2Xdot1ConfigProtocolTable 1 }

L1dpV2Xdot1ConfigProtocolEntry ::= SEQUENCE {
 1ldpV2Xdot1ConfigProtocolTxEnable TruthValue
 }

1ldpV2Xdot1ConfigProtocolTxEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The boolean value that indicates whether the corresponding Local System Protocol Identity instance is transmitted

on the port defined by the given lldpV2Xdot1LocProtocolEntry.

The value of this object is restored from non-volatile storage after a re-initialization of the management system.”

REFERENCE

“9.1.2.1 of IEEE Std 802.1AB”

DEFVAL { false }

::= { lldpV2Xdot1ConfigProtocolEntry 1 }

--

-- lldpV2Xdot1ConfigVidUsageDigestTable: configure the transmission of the
-- VID Usage Digest TLVs on set of ports.

--

lldpV2Xdot1ConfigVidUsageDigestTable OBJECT-TYPE

SYNTAX SEQUENCE OF LldpV2Xdot1ConfigVidUsageDigestEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

“A table that controls selection of LLDP VID Usage Digest TLVs to be transmitted on individual ports.”

::= { lldpV2Xdot1Config 5 }

lldpV2Xdot1ConfigVidUsageDigestEntry OBJECT-TYPE

SYNTAX LldpV2Xdot1ConfigVidUsageDigestEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

“LLDP configuration information that specifies the set of ports (represented as a PortList) on which the local system VID Usage Digest instance will be transmitted.

This configuration object augments the lldpLocVidUsageDigestEntry, therefore it is only present along with the VID Usage Digest instance contained in the associated lldpV2Xdot1LocVidUsageDigestEntry entry.

Each active lldpConfigVidUsageDigestEntry must be restored from non-volatile storage and re-created (along with the corresponding lldpV2Xdot1LocVidUsageDigestEntry) after a re-initialization of the management system.”

AUGMENTS { lldpV2Xdot1LocVidUsageDigestEntry }

::= { lldpV2Xdot1ConfigVidUsageDigestTable 1 }

LldpV2Xdot1ConfigVidUsageDigestEntry ::= SEQUENCE {

lldpV2Xdot1ConfigVidUsageDigestTxEnable TruthValue

}

lldpV2Xdot1ConfigVidUsageDigestTxEnable OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

“The boolean value that indicates whether the corresponding Local System VID Usage Digest instance will be transmitted on the port defined by the given lldpV2Xdot1LocVidUsageDigestEntry. The value of this object must be restored from non-volatile storage after a reinitialization of the management system.”

REFERENCE

“9.1.2.1 of IEEE Std 802.1AB”

DEFVAL { false }

::= { lldpV2Xdot1ConfigVidUsageDigestEntry 1 }

```
--  
-- lldpV2Xdot1ConfigManVidTable : configure the transmission of the  
-- Management VID TLVs on set of ports.  
--  
lldpV2Xdot1ConfigManVidTable OBJECT-TYPE  
    SYNTAX SEQUENCE OF LldpV2Xdot1ConfigManVidEntry  
    MAX-ACCESS not-accessible  
    STATUS current  
    DESCRIPTION  
        "A table that controls selection of LLDP Management VID  
        TLVs to be transmitted on individual ports."  
 ::= { lldpV2Xdot1Config 6 }  
  
lldpV2Xdot1ConfigManVidEntry OBJECT-TYPE  
    SYNTAX LldpV2Xdot1ConfigManVidEntry  
    MAX-ACCESS not-accessible  
    STATUS current  
    DESCRIPTION  
        "LLDP configuration information that specifies the set of  
        port/destination address pairs on which the Local  
        System Management VID will be transmitted.  
        This configuration object augments the  
        lldpV2Xdot1LocManVidEntry, therefore it is  
        only present along with the Management VID contained  
        in the associated lldpV2Xdot1LocManVidEntry entry.  
        Each active lldpV2Xdot1ConfigManVidEntry must be  
        restored from non-volatile storage (along with the  
        corresponding lldpV2Xdot1LocManVidEntry) after a  
        re-initialization of the management system."  
    AUGMENTS { lldpV2Xdot1LocManVidEntry }  
 ::= { lldpV2Xdot1ConfigManVidTable 1 }  
  
LldpV2Xdot1ConfigManVidEntry ::= SEQUENCE {  
    lldpV2Xdot1ConfigManVidTxEnable TruthValue  
}  
  
lldpV2Xdot1ConfigManVidTxEnable OBJECT-TYPE  
    SYNTAX TruthValue  
    MAX-ACCESS read-write  
    STATUS current  
    DESCRIPTION  
        "The lldpV2Xdot1ConfigManVidTxEnable, which is defined as a  
        truth value and configured by the network management,  
        determines whether the IEEE 802.1 organizationally  
        defined Management VID TLV transmission is allowed on a given  
        LLDP transmission capable port.  
        The value of this object must be restored from  
        non-volatile storage after a re-initialization of the  
        management system."  
    REFERENCE  
        "9.1.2.1 of IEEE Std 802.1AB"  
    DEFVAL { false }  
 ::= { lldpV2Xdot1ConfigManVidEntry 1 }
```

```
-- IEEE 802.1 - Local System Information
```

```

--  

-- lldpV2Xdot1LocTable - indexed by ifIndex.  

--  

lldpV2Xdot1LocTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1LocEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one row per port for IEEE 802.1
         organizationally defined LLDP extension on the local system
         known to this agent."
    ::= { lldpV2Xdot1LocalData 1 }

lldpV2Xdot1LocEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1LocEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about IEEE 802.1 organizationally defined
         LLDP extension."
    INDEX     { lldpV2LocPortIfIndex }
    ::= { lldpV2Xdot1LocTable 1 }

LldpV2Xdot1LocEntry ::= SEQUENCE {
    lldpV2Xdot1LocPortVlanId      Unsigned32
}

lldpV2Xdot1LocPortVlanId OBJECT-TYPE
    SYNTAX      Unsigned32(0|1..4094)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The integer value used to identify the port's VLAN identifier
         associated with the local system. A value of zero shall
         be used if the system either does not know the PVID or does
         not support port-based VLAN operation."
    REFERENCE
        "D. 2. 1. 1"
    ::= { lldpV2Xdot1LocEntry 1 }

--  

-- lldpV2Xdot1LocProtoVlanTable: Port and Protocol VLAN information  

-- re-indexed by ifIndex.  

--  

lldpV2Xdot1LocProtoVlanTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1LocProtoVlanEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one or more rows per Port and Protocol
         VLAN information about the local system."
    ::= { lldpV2Xdot1LocalData 2 }

lldpV2Xdot1LocProtoVlanEntry OBJECT-TYPE

```

SYNTAX LldpV2Xdot1LocProtoVlanEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "Port and protocol VLAN ID Information about a particular port component. There may be multiple port and protocol VLANs, identified by a particular lldpV2Xdot1LocProtoVlanId, configured on the given port."
INDEX { lldpV2LocPortIfIndex,
 lldpV2Xdot1LocProtoVlanId }
 ::= { lldpV2Xdot1LocProtoVlanTable 1 }

LldpV2Xdot1LocProtoVlanEntry ::= SEQUENCE {
 lldpV2Xdot1LocProtoVlanId Unsigned32,
 lldpV2Xdot1LocProtoVlanSupported TruthValue,
 lldpV2Xdot1LocProtoVlanEnabled TruthValue
}

lldpV2Xdot1LocProtoVlanId OBJECT-TYPE
 SYNTAX Unsigned32(0|1..4094)
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "The integer value used to identify the port and protocol VLANs associated with the given port associated with the local system. A value of zero shall be used if the system either does not know the protocol VLAN ID (PPVID) or does not support port and protocol VLAN operation."
 REFERENCE
 "D. 2. 2. 2"
 ::= { lldpV2Xdot1LocProtoVlanEntry 1 }

lldpV2Xdot1LocProtoVlanSupported OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The truth value used to indicate whether the given port (associated with the local system) supports port and protocol VLANs."
 REFERENCE
 "D. 2. 2. 1"
 ::= { lldpV2Xdot1LocProtoVlanEntry 2 }

lldpV2Xdot1LocProtoVlanEnabled OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The truth value used to indicate whether the port and protocol VLANs are enabled on the given port associated with the local system."
 REFERENCE
 "D. 2. 2. 1"
 ::= { lldpV2Xdot1LocProtoVlanEntry 3 }

--

```

-- lldpV2Xdot1LocVlanNameTable : VLAN name information about the local system
-- indexed by ifIndex.
--

lldpV2Xdot1LocVlanNameTable OBJECT-TYPE
  SYNTAX      SEQUENCE OF LldpV2Xdot1LocVlanNameEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "This table contains one or more rows per IEEE 802.1Q VLAN
     name information on the local system known to this agent."
 ::= { lldpV2Xdot1LocalData 3 }

lldpV2Xdot1LocVlanNameEntry OBJECT-TYPE
  SYNTAX      LldpV2Xdot1LocVlanNameEntry
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "VLAN name Information about a particular port component.
     There may be multiple VLANs, identified by a particular
     lldpV2Xdot1LocVlanId, configured on the given port."
  INDEX   { lldpV2LocPortIfIndex,
            lldpV2Xdot1LocVlanId }
 ::= { lldpV2Xdot1LocVlanNameTable 1 }

LldpV2Xdot1LocVlanNameEntry ::= SEQUENCE {
  lldpV2Xdot1LocVlanId          VlanId,
  lldpV2Xdot1LocVlanName        SnmpAdminString
}

lldpV2Xdot1LocVlanId OBJECT-TYPE
  SYNTAX      VlanId
  MAX-ACCESS  not-accessible
  STATUS      current
  DESCRIPTION
    "The integer value used to identify the IEEE 802.1Q
     VLAN IDs with which the given port is compatible."
  REFERENCE
    "D. 2. 3. 2"
 ::= { lldpV2Xdot1LocVlanNameEntry 1 }

lldpV2Xdot1LocVlanName OBJECT-TYPE
  SYNTAX      SnmpAdminString (SIZE(1..32))
  MAX-ACCESS  read-only
  STATUS      current
  DESCRIPTION
    "The string value used to identify VLAN name identified by the
     Vlan Id associated with the given port on the local system.

    This object should contain the value of the dot1QVLANStaticName
    object (defined in IETF RFC 4363) identified with the given
    lldpV2Xdot1LocVlanId."
  REFERENCE
    "D. 2. 3. 4"
 ::= { lldpV2Xdot1LocVlanNameEntry 2 }

--

```

```
-- lldpV2Xdot1LocProtocolTable : Protocol Identity information
-- re-indexed by ifIndex and destination address
--

lldpV2Xdot1LocProtocolTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1LocProtocolEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one or more rows per protocol identity
         information on the local system known to this agent."
    REFERENCE
        "D. 2. 4"
    ::= { lldpV2Xdot1LocalData 4 }

lldpV2Xdot1LocProtocolEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1LocProtocolEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Information about particular protocols that are accessible
         through the given port component.

         There may be multiple protocols, identified by particular
         lldpV2Xdot1ProtocolIndex, lldpV2LocPortIfIndex"
    REFERENCE
        "D. 2. 4"
    INDEX      { lldpV2LocPortIfIndex,
                 lldpV2Xdot1LocProtocolIndex }
    ::= { lldpV2Xdot1LocProtocolTable 1 }

LldpV2Xdot1LocProtocolEntry ::= SEQUENCE {
    lldpV2Xdot1LocProtocolIndex Unsigned32,
    lldpV2Xdot1LocProtocolId    OCTET STRING
}

lldpV2Xdot1LocProtocolIndex OBJECT-TYPE
    SYNTAX      Unsigned32(1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object represents an arbitrary local integer value used
         by this agent to identify a particular protocol identity."
    ::= { lldpV2Xdot1LocProtocolEntry 1 }

lldpV2Xdot1LocProtocolId OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (1..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The octet string value used to identify the protocols
         associated with the given port of the local system."
    REFERENCE
        "D. 2. 4. 3"
    ::= { lldpV2Xdot1LocProtocolEntry 2 }

--
-- lldpV2Xdot1LocVidUsageDigestTable: Table of hash values of
```

```

-- system VID Usage Table transmitted
-- via VID Usage Digest TLV.
--


1lldpV2Xdot1LocVidUsageDigestTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1LocVidUsageDigestEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one row per ifIndex/
         destination MAC address pair for usage digest
         information on the local system known to this agent."
    REFERENCE
        "D. 2.5"
    ::= { lldpV2Xdot1LocalData 5 }

1lldpV2Xdot1LocVidUsageDigestEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1LocVidUsageDigestEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Usage digest information to be transmitted
         through the given port."
    REFERENCE
        "D. 2.5"
    INDEX      { lldpV2LocPortIfIndex }
    ::= { lldpV2Xdot1LocVidUsageDigestTable 1 }

LldpV2Xdot1LocVidUsageDigestEntry ::= SEQUENCE {
    lldpV2Xdot1LocVidUsageDigest Unsigned32
}

1lldpV2Xdot1LocVidUsageDigest OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "The integer value obtained by applying the CRC32 function
         to the 128-octet VID Usage Table. A bit of the VID Usage
         Table contains the value PBB-TE-USAGE (binary 1) if the
         corresponding element of the MST Configuration Table
         (IEEE Std 802.1Q 8.9.1) contains the value PBB-TE MSTID (hex FFE)
         and otherwise contains the value NON-PBB-TE-USAGE (binary 0)."
    REFERENCE
        "D. 2.5.1"
    ::= { lldpV2Xdot1LocVidUsageDigestEntry 1 }

-- 
-- 1lldpV2Xdot1LocManVidTable: Table of values configured on the Local
-- system for the Management VID, or the value 0 if a Management VID has not
-- been provisioned.
--


1lldpV2Xdot1LocManVidTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1LocManVidEntry
    MAX-ACCESS  not-accessible
    STATUS      current

```

DESCRIPTION

"This table contains one row per ifIndex/
destination MAC address pair for usage digest
information on the local system known to this agent."

REFERENCE

"D. 2. 6"

::= { lldpV2Xdot1LocalData 6 }

lldpV2Xdot1LocManVidEntry OBJECT-TYPE

SYNTAX LldpV2Xdot1LocManVidEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"Usage digest information to be transmitted
through the given port."

REFERENCE

"D. 2. 6"

INDEX { lldpV2LocPortIfIndex }

::= { lldpV2Xdot1LocManVidTable 1 }

LldpV2Xdot1LocManVidEntry ::= SEQUENCE {

 lldpV2Xdot1LocManVid Unsigned32

}

lldpV2Xdot1LocManVid OBJECT-TYPE

SYNTAX Unsigned32 (0|1..4094)

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The integer value configured on the Local system for
the Management VID, or
the value 0 if a Management VID has not been provisioned."

REFERENCE

"D. 2. 6. 1"

::= { lldpV2Xdot1LocManVidEntry 1 }

-- IEEE 802.1 - Local System Information - Link Aggregation

--- lldpV2Xdot1LocLinkAggTable: Link Aggregation Information Table

lldpV2Xdot1LocLinkAggTable OBJECT-TYPE

SYNTAX SEQUENCE OF LldpV2Xdot1LocLinkAggEntry

MAX-ACCESS not-accessible

STATUS current

DESCRIPTION

"This table contains one row per port of link aggregation
information (as a part of the LLDP 802.1 organizational
extension) on the local system known to this agent."

::= { lldpV2Xdot1LocalData 7 }

lldpV2Xdot1LocLinkAggEntry OBJECT-TYPE

SYNTAX LldpV2Xdot1LocLinkAggEntry

MAX-ACCESS not-accessible

STATUS current
 DESCRIPTION "Link Aggregation information about a particular port component."
 INDEX { lldpV2LocPortIfIndex }
 ::= { lldpV2Xdot1LocLinkAggTable 1 }

LldpV2Xdot1LocLinkAggEntry ::= SEQUENCE {
 lldpV2Xdot1LocLinkAggStatus LldpV2LinkAggStatusMap,
 lldpV2Xdot1LocLinkAggPortId Unsigned32
}

lldpV2Xdot1LocLinkAggStatus OBJECT-TYPE
 SYNTAX LldpV2LinkAggStatusMap
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION "The bitmap value contains the link aggregation capabilities and the current aggregation status of the link."
 REFERENCE "D. 2. 7. 1"
 ::= { lldpV2Xdot1LocLinkAggEntry 1 }

lldpV2Xdot1LocLinkAggPortId OBJECT-TYPE
 SYNTAX Unsigned32(0|1..2147483647)
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION "This object contains the IEEE 802.1 aggregated port identifier, aAggPortID (IEEE Std 802.1AX, 6.3.2.1.1), derived from the ifNumber of the ifIndex for the port component in link aggregation.
 If the port is not in link aggregation state and/or it does not support link aggregation, this value should be set to zero."
 REFERENCE "D. 2. 7. 1"
 ::= { lldpV2Xdot1LocLinkAggEntry 2 }

-- IEEE 802.1 - Remote System Information

--
-- lldpV2Xdot1RemTable - re-indexed for ifIndex and destination MAC address
--

lldpV2Xdot1RemTable OBJECT-TYPE
 SYNTAX SEQUENCE OF LldpV2Xdot1RemEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION "This table contains one or more rows per physical network connection known to this agent. The agent may wish to ensure that only one lldpV2Xdot1RemEntry is present for each local port, or it may choose to maintain multiple

```
    11dpV2Xdot1RemEntries for the same local port."  
    ::= { 11dpV2Xdot1RemoteData 1 }  
  
11dpV2Xdot1RemEntry OBJECT-TYPE  
    SYNTAX      L1dpV2Xdot1RemEntry  
    MAX-ACCESS  not-accessible  
    STATUS      current  
    DESCRIPTION  
        "Information about a particular port component."  
    INDEX      { 11dpV2RemTimeMark,  
                 11dpV2RemLocalIfIndex,  
                 11dpV2RemLocalDestMACAddress,  
                 11dpV2RemIndex }  
    ::= { 11dpV2Xdot1RemTable 1 }  
  
L1dpV2Xdot1RemEntry ::= SEQUENCE {  
    11dpV2Xdot1RemPortVlanId          Unsigned32  
}  
  
11dpV2Xdot1RemPortVlanId OBJECT-TYPE  
    SYNTAX      Unsigned32(0|1..4094)  
    MAX-ACCESS  read-only  
    STATUS      current  
    DESCRIPTION  
        "The integer value used to identify the port's VLAN identifier  
         associated with the remote system. If the remote system  
         either does not know the PVID or does not support port-based  
         VLAN operation, the value of 11dpV2Xdot1RemPortVlanId should  
         be zero."  
    REFERENCE  
        "D. 2. 1. 1"  
    ::= { 11dpV2Xdot1RemEntry 1 }  
  
--  
-- 11dpV2Xdot1RemProtoVlanTable - re-indexed by ifIndex and  
-- destination MAC address  
--  
11dpV2Xdot1RemProtoVlanTable OBJECT-TYPE  
    SYNTAX      SEQUENCE OF L1dpV2Xdot1RemProtoVlanEntry  
    MAX-ACCESS  not-accessible  
    STATUS      current  
    DESCRIPTION  
        "This table contains one or more rows per Port and Protocol  
         VLAN information about the remote system, received on the  
         given port."  
    ::= { 11dpV2Xdot1RemoteData 2 }  
  
11dpV2Xdot1RemProtoVlanEntry OBJECT-TYPE  
    SYNTAX      L1dpV2Xdot1RemProtoVlanEntry  
    MAX-ACCESS  not-accessible  
    STATUS      current  
    DESCRIPTION  
        "Port and protocol VLAN name Information about a particular  
         port component. There may be multiple protocol VLANs,  
         identified by a particular 11dpV2Xdot1RemProtoVlanId, configured  
         on the remote system."
```

```

INDEX { lldpV2RemTimeMark,
        lldpV2RemLocalIfIndex,
        lldpV2RemLocalDestMACAddress,
        lldpV2RemIndex,
        lldpV2Xdot1RemProtoVlanId }
       ::= { lldpV2Xdot1RemProtoVlanTable 1 }

LldpV2Xdot1RemProtoVlanEntry ::= SEQUENCE {
    lldpV2Xdot1RemProtoVlanId          Unsigned32,
    lldpV2Xdot1RemProtoVlanSupported   TruthValue,
    lldpV2Xdot1RemProtoVlanEnabled     TruthValue
}

lldpV2Xdot1RemProtoVlanId OBJECT-TYPE
SYNTAX      Unsigned32(0|1..4094)
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "The integer value used to identify the port and protocol
    VLANs associated with the given port associated with the
    remote system.

    If port and protocol VLANs are not supported on the given
    port associated with the remote system, or if the port is
    not enabled with any port and protocol VLAN, the value of
    lldpV2Xdot1RemProtoVlanId should be zero."
REFERENCE
    "D. 2. 2. 2"
 ::= { lldpV2Xdot1RemProtoVlanEntry 1 }

lldpV2Xdot1RemProtoVlanSupported OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The truth value used to indicate whether the given port
    (associated with the remote system) is capable of supporting
    port and protocol VLANs."
REFERENCE
    "D. 2. 2. 1"
 ::= { lldpV2Xdot1RemProtoVlanEntry 2 }

lldpV2Xdot1RemProtoVlanEnabled OBJECT-TYPE
SYNTAX      TruthValue
MAX-ACCESS  read-only
STATUS      current
DESCRIPTION
    "The truth value used to indicate whether the port and
    protocol VLANs are enabled on the given port associated with
    the remote system."
REFERENCE
    "D. 2. 2. 1"
 ::= { lldpV2Xdot1RemProtoVlanEntry 3 }

--  

-- lldpV2Xdot1RemVlanNameTable : VLAN name information of the remote  

-- systems

```

```
-- Re-indexed by ifIndex and destination MAC address
--  
  
l1dpV2Xdot1RemVlanNameTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF L1dpV2Xdot1RemVlanNameEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one or more rows per IEEE 802.1Q VLAN
         name information about the remote system, received on the
         given port."
    REFERENCE
        "D. 2. 3"
    ::= { l1dpV2Xdot1RemoteData 3 }  
  
l1dpV2Xdot1RemVlanNameEntry OBJECT-TYPE
    SYNTAX      L1dpV2Xdot1RemVlanNameEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "VLAN name Information about a particular port component.
         There may be multiple VLANs, identified by a particular
         l1dpV2Xdot1RemVlanId, received on the given port."
    INDEX     { l1dpV2RemTimeMark,
                l1dpV2RemLocalIfIndex,
                l1dpV2RemLocalDestMACAddress,
                l1dpV2RemIndex,
                l1dpV2Xdot1RemVlanId }
    ::= { l1dpV2Xdot1RemVlanNameTable 1 }  
  
L1dpV2Xdot1RemVlanNameEntry ::= SEQUENCE {
        l1dpV2Xdot1RemVlanId      VlanId,
        l1dpV2Xdot1RemVlanName    SnmpAdminString
    }  
  
l1dpV2Xdot1RemVlanId OBJECT-TYPE
    SYNTAX      VlanId
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The integer value used to identify the IEEE 802.1Q
         VLAN IDs with which the given port of the remote system
         is compatible."
    REFERENCE
        "D. 2. 3. 2"
    ::= { l1dpV2Xdot1RemVlanNameEntry 1 }  
  
l1dpV2Xdot1RemVlanName OBJECT-TYPE
    SYNTAX      SnmpAdminString (SIZE(1.. 32))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The string value used to identify VLAN name identified by the
         VLAN Id associated with the remote system."
    REFERENCE
        "D. 2. 3. 4"
    ::= { l1dpV2Xdot1RemVlanNameEntry 2 }
```

```

--  

-- lldpV2Xdot1RemProtocolTable : Protocol information of the remote systems  

-- Re-indexed by ifIndex and destination MAC address  

--  

lldpV2Xdot1RemProtocolTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1RemProtocolEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one or more rows per protocol information
         about the remote system, received on the given port."
    ::= { lldpV2Xdot1RemoteData 4 }

lldpV2Xdot1RemProtocolEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1RemProtocolEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Protocol information about a particular port component.
         There may be multiple protocols, identified by a particular
         lldpV2Xdot1ProtocolIndex, received on the given port."
    INDEX      { lldpV2RemTimeMark,
                 lldpV2RemLocalIfIndex,
                 lldpV2RemLocalDestMACAddress,
                 lldpV2RemIndex,
                 lldpV2Xdot1RemProtocolIndex }
    ::= { lldpV2Xdot1RemProtocolTable 1 }

LldpV2Xdot1RemProtocolEntry ::= SEQUENCE {
    lldpV2Xdot1RemProtocolIndex      Unsigned32,
    lldpV2Xdot1RemProtocolId        OCTET STRING
}

lldpV2Xdot1RemProtocolIndex OBJECT-TYPE
    SYNTAX      Unsigned32(1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This object represents an arbitrary local integer value used
         by this agent to identify a particular protocol identity."
    ::= { lldpV2Xdot1RemProtocolEntry 1 }

lldpV2Xdot1RemProtocolId OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE (1..255))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The octet string value used to identify the protocols
         associated with the given port of remote system."
    REFERENCE
        "D. 2. 4. 3"
    ::= { lldpV2Xdot1RemProtocolEntry 2 }

--
```

```
-- lldpV2Xdot1RemVidUsageDigestTable: Table of hash values of
-- system VID Usage Table received
-- via VID Usage Digest TLV.
--  
  
lldpV2Xdot1RemVidUsageDigestTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1RemVidUsageDigestEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains one row per ifIndex/
         destination MAC address pair for usage digest
         information received by the local system."
    REFERENCE
        "D. 2.5"
    ::= { lldpV2Xdot1RemoteData 5 }  
  
lldpV2Xdot1RemVidUsageDigestEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1RemVidUsageDigestEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Usage digest information received on
         the given port/destination address pair."
    REFERENCE
        "D. 2.5"
    INDEX      { lldpV2RemTimeMark,
                  lldpV2RemLocalIfIndex,
                  lldpV2RemLocalDestMACAddress }
    ::= { lldpV2Xdot1RemVidUsageDigestTable 1 }  
  
LldpV2Xdot1RemVidUsageDigestEntry ::= SEQUENCE {
    lldpV2Xdot1RemVidUsageDigest  Unsigned32
}  
  
lldpV2Xdot1RemVidUsageDigest OBJECT-TYPE
    SYNTAX Unsigned32
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "The integer value obtained by applying the CRC32 function
         to the 128-octet VID Usage Table. A bit of the VID Usage
         Table contains the value PBB-TE-USAGE (binary 1) if the
         corresponding element of the MST Configuration Table
         (IEEE Std 802.1Q 8.9.1) contains the value PBB-TE MSTID (hex FFE)
         and otherwise contains the value NON-PBB-TE-USAGE (binary 0)."
    REFERENCE
        "D. 2.5.1"
    ::= { lldpV2Xdot1RemVidUsageDigestEntry 1 }  
  
--  
-- lldpV2Xdot1RemManVidTable: Table of values configured on remote
-- systems for the Management VID, or the value 0 if a Management VID has not
-- been provisioned.  
--  
lldpV2Xdot1RemManVidTable OBJECT-TYPE
```

SYNTAX SEQUENCE OF LldpV2Xdot1RemManVidEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
 "This table contains one row per ifIndex/
 destination MAC address pair for management VID
 information received from remote systems."
REFERENCE
 "D. 2. 6"
 ::= { lldpV2Xdot1RemoteData 6 }

```
l1dpV2Xdot1RemManVidEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1RemManVidEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Management VID information received
         through the given port/destination address pair."
```

```

REFERENCE      "D. 2. 6"
INDEX          { 11dpV2RemTimeMark,
                  11dpV2RemLocalIfIndex,
                  11dpV2RemLocalDestMACAddress }
 ::= { 11dpV2Xdot1RemManVidTable 1 }

```

```
LldpV2Xdot1RemManVidEntry ::= SEQUENCE {
    lldpV2Xdot1RemManVid          Unsigned32
}
```

```
11dpV2Xdot1RemManVid OBJECT-TYPE
    SYNTAX Unsigned32 (0|1..4094)
    MAX-ACCESS read-only
    STATUS current
    DESCRIPTION
        "The integer value configured on a system for
         the Management VID, or
         the value 0 if a Management VID has not been provisioned."
    REFERENCE
        "D. 2. 6. 1"
::= { 11dpV2Xdot1RemManVidEntry 1 }
```

-- Remote System Information - Link Aggregation

11.d.v2v.dat1RawLinkAveTable : Link Aggregation Information Table

```
1lldpV2Xdot1RemLinkAggTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF LldpV2Xdot1RemLinkAggEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "This table contains port link aggregation information
         (as a part of the LLDP IEEE 802.1 organizational extension)
```

```
        of the remote system."
 ::= { lldpV2Xdot1RemoteData 7 }

lldpV2Xdot1RemLinkAggEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1RemLinkAggEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Link Aggregation information about remote system's port
         component."
    INDEX      { lldpV2RemTimeMark,
                  lldpV2RemLocalIfIndex,
                  lldpV2RemLocalDestMACAddress,
                  lldpV2RemIndex }
 ::= { lldpV2Xdot1RemLinkAggTable 1 }

LldpV2Xdot1RemLinkAggEntry ::= SEQUENCE {
    lldpV2Xdot1RemLinkAggStatus          LldpV2LinkAggStatusMap,
    lldpV2Xdot1RemLinkAggPortId         Unsigned32
}

lldpV2Xdot1RemLinkAggStatus OBJECT-TYPE
    SYNTAX      LldpV2LinkAggStatusMap
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The bitmap value contains the link aggregation capabilities
         and the current aggregation status of the link."
    REFERENCE
        "D. 2. 7. 1"
 ::= { lldpV2Xdot1RemLinkAggEntry 1 }

lldpV2Xdot1RemLinkAggPortId OBJECT-TYPE
    SYNTAX      Unsigned32(0|1..2147483647)
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the IEEE 802.1 aggregated port
         identifier, aAggPortID (IEEE Std 802.1AX, 6.3.2.1.1),
         derived from the ifNumber of the ifIndex for the port
         component associated with the remote system.

         If the remote port is not in link aggregation state and/or
         it does not support link aggregation, this value should be
         zero."
    REFERENCE
        "D. 2. 7. 1"
 ::= { lldpV2Xdot1RemLinkAggEntry 2 }
```

-- Conformance Information for the basicSet TLV set

```
lldpV2Xdot1Conformance OBJECT IDENTIFIER ::= { lldpV2Xdot1MIB 2 }
lldpV2Xdot1Compliances OBJECT IDENTIFIER ::= { lldpV2Xdot1Conformance 1 }
lldpV2Xdot1Groups      OBJECT IDENTIFIER ::= { lldpV2Xdot1Conformance 2 }
```

-- compliance statements

```

11dpV2Xdot1TxRxCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    "A compliance statement for SNMP entities that implement
     the IEEE 802.1 organizationally defined LLDP extension MIB.

This group is mandatory for all agents that implement the
LLDP 802.1 organizational extension in TX and/or RX mode
for the basicSet TLV set.

This version defines compliance requirements for
V2 of the LLDP MIB."
MODULE -- this module
  MANDATORY-GROUPS { 11dpV2Xdot1ConfigGroup,
                     ifGeneralInformationGroup
  }
 ::= { 11dpV2Xdot1Compliances 1 }

11dpV2Xdot1TxCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    "A compliance statement for SNMP entities that implement
     the IEEE 802.1 organizationally defined LLDP extension MIB.

This group is mandatory for agents that implement the
LLDP 802.1 organizational extension in the RX mode
for the basicSet TLV set.

This version defines compliance requirements for
V2 of the LLDP MIB."
MODULE -- this module
  MANDATORY-GROUPS { 11dpV2Xdot1LocSysGroup }

 ::= { 11dpV2Xdot1Compliances 2 }

11dpV2Xdot1RxCompliance MODULE-COMPLIANCE
  STATUS current
  DESCRIPTION
    "A compliance statement for SNMP entities that implement
     the IEEE 802.1 organizationally defined LLDP extension MIB.

This group is mandatory for agents that implement the
LLDP 802.1 organizational extension in the RX mode
for the basicSet TLV set.

This version defines compliance requirements for
V2 of the LLDP MIB."
MODULE -- this module
  MANDATORY-GROUPS { 11dpV2Xdot1RemSysGroup }

 ::= { 11dpV2Xdot1Compliances 3 }

-- MIB groupings for the basicSet TLV set

11dpV2Xdot1ConfigGroup   OBJECT-GROUP
  OBJECTS {

```

```
    lldpV2Xdot1ConfigPortVlanTxEnable,
    lldpV2Xdot1ConfigVlanNameTxEnable,
    lldpV2Xdot1ConfigProtoVlanTxEnable,
    lldpV2Xdot1ConfigProtocolTxEnable,
    lldpV2Xdot1ConfigVidUsageDigestTxEnable,
    lldpV2Xdot1ConfigManVidTxEnable
}
STATUS current
DESCRIPTION
    "The collection of objects which are used to configure the
    IEEE 802.1 organizationally defined LLDP extension
    implementation behavior for the basicSet TLV set."
 ::= { lldpV2Xdot1Groups 1 }

lldpV2Xdot1LocSysGroup OBJECT-GROUP
OBJECTS {
    lldpV2Xdot1LocPortVlanId,
    lldpV2Xdot1LocProtoVlanSupported,
    lldpV2Xdot1LocProtoVlanEnabled,
    lldpV2Xdot1LocVlanName,
    lldpV2Xdot1LocProtocolId,
    lldpV2Xdot1LocVidUsageDigest,
    lldpV2Xdot1LocManVid,
    lldpV2Xdot1LocLinkAggStatus,
    lldpV2Xdot1LocLinkAggPortId
}
STATUS current
DESCRIPTION
    "The collection of objects which are used to represent
    IEEE 802.1 organizationally defined LLDP extension associated
    with the Local Device Information for the basicSet TLV set."
 ::= { lldpV2Xdot1Groups 2 }

lldpV2Xdot1RemSysGroup OBJECT-GROUP
OBJECTS {
    lldpV2Xdot1RemPortVlanId,
    lldpV2Xdot1RemProtoVlanSupported,
    lldpV2Xdot1RemProtoVlanEnabled,
    lldpV2Xdot1RemVlanName,
    lldpV2Xdot1RemProtocolId,
    lldpV2Xdot1RemVidUsageDigest,
    lldpV2Xdot1RemManVid,
    lldpV2Xdot1RemLinkAggStatus,
    lldpV2Xdot1RemLinkAggPortId
}
STATUS current
DESCRIPTION
    "The collection of objects which are used to represent LLDP
    802.1 organizational extension Remote Device Information
    for the basicSet TLV set."
 ::= { lldpV2Xdot1Groups 3 }
```

```
-- Organizationally Defined Information Extension - IEEE 802.1
-- Definitions to support the cnSet TLV set (Table D-1)
-- for Congestion Notification
--
```

```

l1dpXdot1CnMIB OBJECT IDENTIFIER ::= { l1dpV2Xdot1MIB 3 }
l1dpXdot1CnObjects OBJECT IDENTIFIER ::= { l1dpXdot1CnMIB 1 }

-- CN 802.1 MIB Extension groups

l1dpXdot1CnConfig OBJECT IDENTIFIER ::= { l1dpXdot1CnObjects 1 }
l1dpXdot1CnLocalData OBJECT IDENTIFIER ::= { l1dpXdot1CnObjects 2 }
l1dpXdot1CnRemoteData OBJECT IDENTIFIER ::= { l1dpXdot1CnObjects 3 }

```

```
-- Textual conventions for Congestion Notification
```

```

L1dpV2CnBitVector ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "This TC describes a bit vector used in the Congestion
         Notification objects. Each bit represents a Boolean status
         associated with a priority code point. A bit value of 0
         represents FALSE, 1 represents TRUE.

        The bit 'pri0status(0)' indicates the status for priority 0
        The bit 'pri1status(1)' indicates the status for priority 1
        The bit 'pri2status(2)' indicates the status for priority 2
        The bit 'pri3status(3)' indicates the status for priority 3
        The bit 'pri4status(4)' indicates the status for priority 4
        The bit 'pri5status(5)' indicates the status for priority 5
        The bit 'pri6status(6)' indicates the status for priority 6
        The bit 'pri7status(7)' indicates the status for priority 7"

```

```

SYNTAX BITS {
    pri0status(0),
    pri1status(1),
    pri2status(2),
    pri3status(3),
    pri4status(4),
    pri5status(5),
    pri6status(6),
    pri7status(7)
}

```

```
-- IEEE 802.1 - Congestion Notification Configuration
```

```

-- l1dpXdot1CnConfigCnTable : configure the
-- transmission of the Congestion Notification TLV on a set of ports
--
```

```

l1dpXdot1CnConfigCnTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF L1dpXdot1CnConfigCnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that controls selection of Congestion Notification"

```

TLVs to be transmitted on individual ports."
 ::= { lldpXdot1CnConfig 1 }

lldpXdot1CnConfigCnEntry OBJECT-TYPE
 SYNTAX LldpXdot1CnConfigCnEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "LLDP configuration information that controls the
 transmission of IEEE 802.1 organizationally defined
 Congestion Notification TLV on LLDP transmission capable ports.

This configuration object augments the lldpV2PortConfigEntry of
 the LLDP-MIB, therefore it is only present along with the port
 configuration defined by the associated lldpV2PortConfigEntry
 entry.

Each active lldpConfigEntry is restored from non-volatile
 storage (along with the corresponding lldpV2PortConfigEntry)
 after a re-initialization of the management system."
 AUGMENTS { lldpV2PortConfigEntry }
 ::= { lldpXdot1CnConfigCnTable 1 }

LldpXdot1CnConfigCnEntry ::= SEQUENCE {
 lldpXdot1CnConfigCnTxEnable TruthValue
 }

lldpXdot1CnConfigCnTxEnable OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-write
 STATUS current
 DESCRIPTION
 "The lldpXdot1CnConfigCnTxEnable, which is
 defined as a truth value and configured by the network
 management, determines whether the IEEE 802.1 organizationally
 defined Congestion Notification TLV transmission is allowed on a
 given LLDP transmission capable port.

The value of this object is restored from non-volatile
 storage after a re-initialization of the management system."
 REFERENCE
 "D. 2.8"
 DEFVAL { false }
 ::= { lldpXdot1CnConfigCnEntry 1 }

-- IEEE 802.1 - Congestion Notification Local System Information

--- lldpV2Xdot1LocCnTable: Port Extension Information Table

lldpV2Xdot1LocCnTable OBJECT-TYPE
 SYNTAX SEQUENCE OF LldpV2Xdot1LocCnEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

"This table contains one row per port of Congestion Notification information (as a part of the LLDP 802.1 organizational extension) on the local system known to this agent."

::= { lldpXdot1CnLocalData 1 }

```

11dpV2Xdot1LocCnEntry OBJECT-TYPE
    SYNTAX      11dpV2Xdot1LocCnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Congestion Notification information about a
         particular port component."
    INDEX      { 11dpV2LocPortIfIndex }
    ::= { 11dpV2Xdot1LocCnTable 1 }

L1dpV2Xdot1LocCnEntry ::= SEQUENCE {
    11dpV2Xdot1LocCnPvIndicators      L1dpV2CnBitVector,
    11dpV2Xdot1LocReadyIndicators    L1dpV2CnBitVector
}

```

```

l1dpV2Xdot1LocCNPVIndicators OBJECT-TYPE
    SYNTAX      L1dpV2CnBitVector
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the CNPV indicators
         for the Port."
    REFERENCE
        "D. 2. 8. 3"
 ::= { l1dpV2Xdot1LocCnEntry 1 }

```

```
11dpV2Xdot1LocReadyIndicators OBJECT-TYPE
    SYNTAX      LldpV2CnBitVector
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the Ready indicators
         for the Port."
    REFERENCE
        "D. 2. 8. 4"
    ::= { 11dpV2Xdot1LocCpEntry 2 }
```

-- IEEE 802.1 - Congestion Notification Remote System Information

---- 11dpV2Xdot1RemCnTable: Port Extension Information Table

1lldpV2Xdot1RemCnTable OBJECT-TYPE
 SYNTAX SEQUENCE OF LldpV2Xdot1RemCnEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
 "This table contains Congestion Notification information
 (as a part of the LLDP IEEE 802.1 organizational extension)

```

        of the remote system."
 ::= { lldpXdot1CnRemoteData 1 }

lldpV2Xdot1RemCnEntry OBJECT-TYPE
    SYNTAX      LldpV2Xdot1RemCnEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "Port Extension information about remote systems port
         component."
    INDEX      { lldpV2RemTimeMark,
                  lldpV2RemLocalIfIndex,
                  lldpV2RemLocalDestMACAddress,
                  lldpV2RemIndex }
 ::= { lldpV2Xdot1RemCnTable 1 }

LldpV2Xdot1RemCnEntry ::= SEQUENCE {
    lldpV2Xdot1RemCNPVIndicators      LldpV2CnBitVector,
    lldpV2Xdot1RemReadyIndicators    LldpV2CnBitVector
}

lldpV2Xdot1RemCNPVIndicators OBJECT-TYPE
    SYNTAX      LldpV2CnBitVector
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the CNPV indicators
         for the Port."
    REFERENCE
        "D. 2. 8. 3"
 ::= { lldpV2Xdot1RemCnEntry 1 }

lldpV2Xdot1RemReadyIndicators OBJECT-TYPE
    SYNTAX      LldpV2CnBitVector
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "This object contains the Ready indicators
         for the Port."
    REFERENCE
        "D. 2. 8. 4"
 ::= { lldpV2Xdot1RemCnEntry 2 }

```

-- IEEE 802.1 - Congestion Notification Conformance Information

```

lldpXdot1CnConformance OBJECT IDENTIFIER ::= { lldpV2Xdot1MIB 4 }

lldpXdot1CnCompliances OBJECT IDENTIFIER ::= { lldpXdot1CnConformance 1 }
lldpXdot1CnGroups OBJECT IDENTIFIER ::= { lldpXdot1CnConformance 2 }

--
```

```

-- Congestion Notification - Compliance Statements
--
```

```

lldpXdot1CnCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION

```

"A compliance statement for SNMP entities that implement the IEEE 802.1 organizationally defined Congestion Notification LLDP extension MIB.

This group is mandatory for agents that implement the Congestion Notification cnSet TLV set."

```
MODULE      -- this module
MANDATORY-GROUPS { lldpXdot1CnGroup,
                   ifGeneralInformationGroup }
 ::= { lldpXdot1CnCompliances 1 }
```

```
--  
-- Congestion Notification - MIB groupings  
--
```

```
lldpXdot1CnGroup OBJECT-GROUP
OBJECTS {
    lldpXdot1CnConfigCnTxEnable,
    lldpV2Xdot1LocCNPVIndicators,
    lldpV2Xdot1LocReadyIndicators,
    lldpV2Xdot1RemCNPVIndicators,
    lldpV2Xdot1RemReadyIndicators
}
STATUS current
DESCRIPTION
    "The collection of objects that support the
     Congestion Notification cnSet TLV set."
 ::= { lldpXdot1CnGroups 1 }
```

END

D.5 PICS proforma for IEEE 802.1 Organizationally Specific TLV extensions⁵¹**D.5.1 Implementation identification**

Supplier	
Contact point for queries about the PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names	
<p>NOTE 1—Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification.</p> <p>NOTE 2—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model).</p>	

D.5.2 Protocol summary, IEEE Std 802.1Q

Identification of protocol specification	IEEE Std 802.1Q-2010, IEEE Standards for Local and metropolitan area networks—Virtual Bridged Local Area Networks	
Identification of amendments and corrigenda to the PICS proforma that have been completed as part of the PICS	Amd. : Corr. :	Amd. : Corr. :
Have any Exception items been required? (See A.3.3: The answer Yes means that the implementation does not conform to IEEE Std 802.1Q-2010.)	No []	Yes []

Date of Statement	
-------------------	--

⁵¹Instructions for completing the PICS Proforma are given in A.3.

D.5.3 Major capabilities and options

Item	Feature	Status	References	Support
dot1basicSet	Is the IEEE 802.1 Organizationally Specific TLV basicSet implemented?	O.1	D.1, Table D-1	Yes [] No []
dot1basictlv	Is each TLV in the IEEE 802.1 Organizationally Specific TLV basicSet implemented? Port VLAN ID TLV Port And Protocol VLAN ID TLV VLAN Name TLV Protocol Identity TLV VID Usage Digest TLV Management VID TLV Link Aggregation TLV	dot1basicSet:M dot1basicSet:M dot1basicSet:M dot1basicSet:M dot1basicSet:M dot1basicSet:M dot1basicSet:M dot1basicSet:M	D.2.1 D.2.2 D.2.3 D.2.4 D.2.5 D.2.6 D.2.7	Yes [] Yes []
dot1cnSet	Is the IEEE 802.1 Organizationally Specific TLV cnSet implemented?	O.1	D.1, Table D-1	Yes [] No []
dot1cntlv	Is each TLV in the IEEE 802.1 Organizationally Specific TLV lagSet implemented? Link Aggregation TLV	dot1cnSet:M	D.2.7	Yes []
lldpmib	Which type of MIB is implemented (one is mandatory)? SNMP is supported (if yes, answer item snmpmib and skip equivstor) SNMP is not supported (if yes, answer equivstor and skip snmpmib)	O.2 O.2	D.4 D.4	Yes [] No [] Yes [] No []
snmpmib	If the SNMP MIB is implemented, is the MIB module in conformance with the MIB sections indicated in Table D-4 for the operating mode being implemented?	M	D.4	Yes []
equivstor	If the SNMP is not supported, is the provided storage and retrieval capability functionally equivalent with the indicated specifications of this clause for the operating mode being implemented?	M	D.2.1, D.2.2, D.2.3, D.2.4, and D.2.7	Yes []

Annex E

(normative)

Notational conventions used in state diagrams

State diagrams are used to represent the operation of the protocol by a number of cooperating state machines each comprising a group of connected, mutually exclusive states. Only one state of each machine can be active at any given time.

Each state is represented in the state diagram as a rectangular box, divided into two parts by a horizontal line. The upper part contains the state identifier, written in upper case letters. The lower part contains any procedures that are executed on entry to the state.

All permissible transitions between states are represented by arrows, the arrowhead denoting the direction of the possible transition. Labels attached to arrows denote the condition(s) that must be met in order for the transition to take place. All conditions are expressions that evaluate to TRUE or FALSE; if a condition evaluates to TRUE, then the condition is met. The label UCT denotes an unconditional transition (i.e., UCT always evaluates to TRUE). A transition that is global in nature (i.e., a transition that occurs from any of the possible states if the condition attached to the arrow is met) is denoted by an open arrow, i.e., no specific state is identified as the origin of the transition. When the condition associated with a global transition is met, it supersedes all other exit conditions including UCT. The special global condition BEGIN supersedes all other global conditions, and once asserted remains asserted until all state blocks have executed to the point that variable assignments and other consequences of their execution remain unchanged.

On entry to a state, the procedures defined for the state (if any) are executed exactly once, in the order that they appear on the page. Each action is deemed to be atomic, i.e., execution of a procedure completes before the next sequential procedure starts to execute. No procedures execute outside of a state block. The procedures in only one state block execute at a time, even if the conditions for execution of state blocks in different state machines are satisfied. All procedures in an executing state block complete execution before the transition to and execution of any other state block occurs, i.e., the execution of any state block appears to be atomic with respect to the execution of any other state block and the transition condition to that state from the previous state is TRUE when execution commences. The order of execution of state blocks in different state machines is undefined except as constrained by their transition conditions. A variable that is set to a particular value in a state block retains this value until a subsequent state block executes a procedure that modifies the value.

On completion of all of the procedures within a state, all exit conditions for the state (including all conditions associated with global transitions) are evaluated continuously until one of the conditions is met. The label ELSE denotes a transition that occurs if none of the other conditions for transitions from the state are met (i.e., ELSE evaluates to TRUE if all other possible exit conditions from the state evaluate to FALSE). Where two or more exit conditions with the same level of precedence become TRUE simultaneously, the choice as to which exit condition causes the state transition to take place is arbitrary.

Where it is necessary to split a state machine description across more than one diagram, a transition between two states that appear on different diagrams is represented by an exit arrow drawn with dashed lines, plus a reference to the diagram that contains the destination state. Similarly, dashed arrows and a dashed state box are used on the destination diagram to show the transition to the destination state. In a state machine that has been split in this way, any global transitions that can cause entry to states defined in one of the diagrams are deemed to be potential exit conditions for all of the states of the state machine, regardless of which diagram the state boxes appear in.

Should a conflict exist between the interpretation of a state diagram and either the corresponding global transition tables or the textual description associated with the state machine, the state diagram takes precedence. The interpretation of the special symbols and operators used in the state diagrams is as defined in Table E-6; these symbols and operators are derived from the notation of the “C++” programming language, ISO/IEC 14882. If a Boolean variable is described in this clause as being set it has or is assigned the value TRUE, if reset or clear the value FALSE.

Table E-6—State machine symbols

Symbol	Interpretation
() ; = !	Used to force the precedence of operators in Boolean expressions and to delimit the argument(s) of actions within state boxes. Used as a terminating delimiter for actions within state boxes. Where a state box contains multiple actions, the order of execution follows the normal English language conventions for reading text. Assignment action. The value of the expression to the right of the operator is assigned to the variable to the left of the operator. Where this operator is used to define multiple assignments, e.g., $a = b = X$ the action causes the value of the expression following the right-most assignment operator to be assigned to all of the variables that appear to the left of the right-most assignment operator. Logical NOT operator.
&& if...then... !=	Logical AND operator. Logical OR operator. Conditional action. If the Boolean expression following the if evaluates to TRUE, then the action following the then is executed. Inequality. Evaluates to TRUE if the expression to the left of the operator is not equal in value to the expression to the right.
== * -	Equality. Evaluates to TRUE if the expression to the left of the operator is equal in value to the expression to the right. Arithmetic multiplication operator. Arithmetic subtraction operator.

Annex F

(informative)

Shared and Independent VLAN Learning

This standard provides for a variety of approaches to the implementation of Bridges from the point of view of the way that individual MAC Addresses are learned, and how that learned information is used in subsequent forwarding/filtering decisions. Two mechanisms are used as a basis for these variants:

- a) Making use of address information learned across a number of VIDs in order to make learning decisions relative to any one of those VIDs. This is referred to as *Shared VLAN Learning* (SVL, 3.165);
- b) Making use of address information learned in association with one VID only in order to make learning decisions relative to that VID, and ensuring that it is not used in learning decisions relative to any other VID. This is referred to as *Independent VLAN Learning* (IVL, 3.75).

These mechanisms lead to the SVL/IVL model for how a Bridge implements learning and filtering for MAC Addresses. Using the terminology of 8.8.8, an SVL/IVL Bridge supports multiple FIDs (which effectively equates to supporting multiple Filtering Databases), and multiple VIDs can use each FID. By varying the number of FIDs supported, and the number of VIDs that can share each FID, the following simplifications of the SVL/IVL model can be created:

- c) *Shared VLAN Learning (SVL) only.* The implementation supports only one FID, so all VIDs share the same learned MAC Address information, regardless of which VID the information was learned in;
- d) *Independent VLAN Learning (IVL) only.* Multiple FIDs are supported, but each FID can support only one VID, so each VID makes use only of MAC Address information learned within the context of that VID.

All three approaches are permitted by this standard, and each has advantages in particular circumstances. The remainder of this annex discusses

- e) The requirements for Independent VLAN Learning, Shared VLAN Learning, or both;
- f) How Bridges are made aware of the requirement for particular VLANs to be “shared” or “independent”;
- g) How Bridges based on one of these models can interoperate with Bridges based on a different model, in the same network.

F.1 Requirements for Shared and Independent Learning

Under most circumstances, the SVL and IVL approaches work equally well, and Bridges adopting either approach can be freely intermixed within a network. There are, however, a small number of configuration cases where, in order to prevent undue flooding of unicast frames, and in some cases, to make communication between the affected end systems possible, it is necessary to make specific choices as to how Bridges that adopt these different learning models are deployed in a network. The following subclauses give examples of some of these configurations and provide a generic statement of the requirements that must be met in order for each learning model to be successfully deployed.

F.1.1 Connecting independent VLANs

Figure F-1 illustrates how a device that connects two VLANs together, and which therefore itself shares learning between those VLANs, creates a need for those VLANs to be independent in other Bridges.

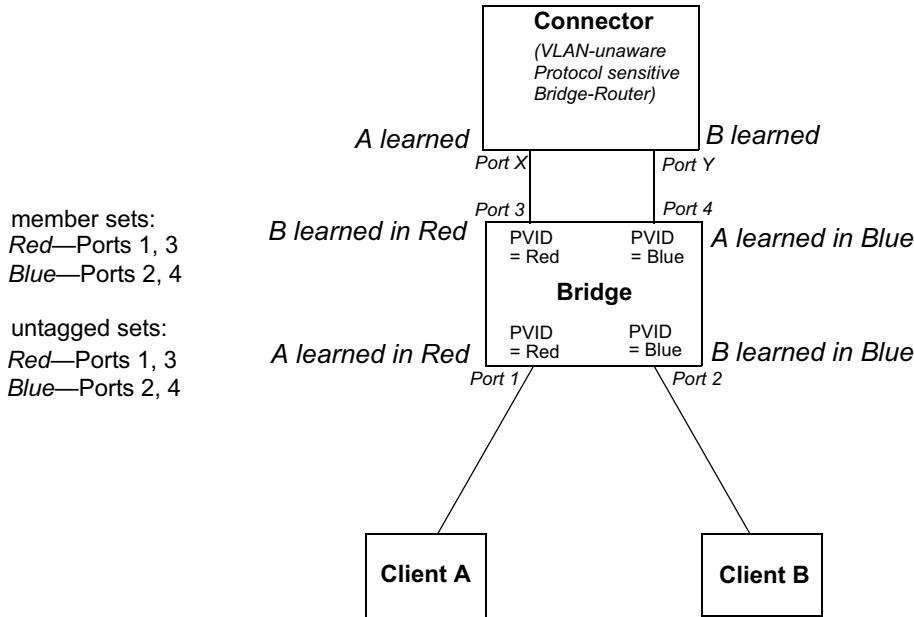


Figure F-1—Connecting independent VLANs—1

Clients A and B are connected via the protocol sensitive Bridge-Router (Connector), with an intervening VLAN-aware Bridge. The fact that all Ports of the Bridge carry untagged traffic neatly conceals the fact that the Connector has the effect of connecting VLANs Red and Blue together with regard to bridged traffic. The Connector learns A and B in the same database, as it has no knowledge of VLANs Red and Blue. This prevents any traffic transmitted on the Red VLAN (Port X of the Connector) that is destined for A, from being bridged to Port Y and transmitted on the Blue VLAN.

The VLAN-aware Bridge must keep its learning separate for Red and Blue; otherwise, the addresses of the two clients would be alternately learned on diagonally opposite Ports as, for example, traffic sourced by A reenters the Bridge on Port 4 having previously been seen on Port 1.

NOTE—This example assumes that Spanning Tree is disabled in the Connector, so that the VLAN-aware Bridge does not attempt to suppress the loop that apparently exists if VLANs are not taken into account.

A simpler example can be constructed, with a single Port connecting the Connector and the VLAN-aware Bridge, if the Connector is itself VLAN-aware and transmits and receives only VLAN-tagged traffic. In this case, the Connector would allocate a single FID for use by Red and Blue. This is shown in Figure F-2.

F.1.2 Duplicate MAC Addresses

The simplest example of a need for Independent VLAN Learning occurs where two (or more) distinct devices in different parts of the network reuse the same individual MAC Address, or where a single device is connected to multiple LANs, and all of its LAN interfaces use the same individual MAC Address. This is shown in Figure F-3.

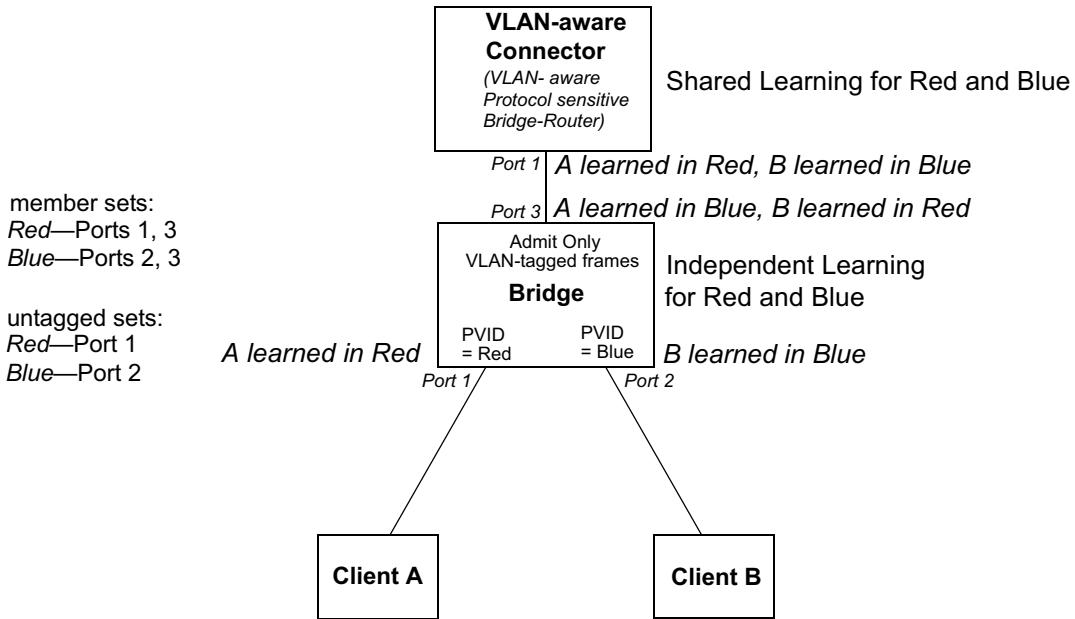


Figure F-2—Connecting independent VLANs—2

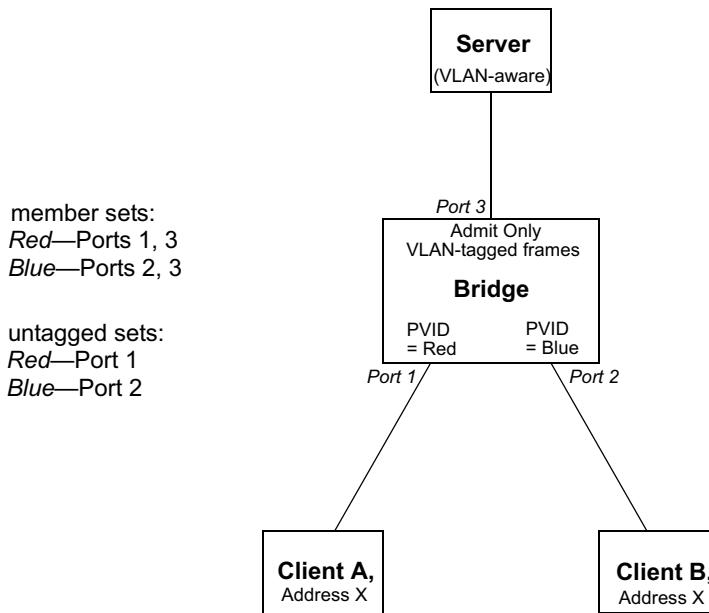


Figure F-3—Duplicate MAC Addresses

The example shows two clients with access to the same server; both clients are using the same individual MAC Address, X. If the Bridge shares learning between VLAN Red (which serves Client A) and VLAN Blue (which serves Client B), i.e., the Bridge uses the same FID for both VIDs, then Address X will appear to move between Ports 1 and 2 of the Bridge, depending on which client has most recently transmitted a frame. Communication between these Clients and the server will therefore be seriously disrupted. Assignment of distinct FIDs for Red and Blue ensures that communication can take place correctly.

Hence, in order to construct this particular VLAN configuration, either an IVL Bridge or an SVL/IVL Bridge would be required.

F.1.3 Asymmetric VLANs and Rooted-Multipoint connectivity

A primary example of the requirement for Shared VLAN Learning is found in “asymmetric” uses of VLANs. Under normal circumstances, a pair of devices communicating in a VLAN environment will both send and receive using the same VID; however, there are some circumstances in which it is convenient to make use of two distinct VIDs, one used for A to transmit to B and the other used for B to transmit to A. Examples of such applications include configuring multi-netted servers, and configuring rooted-multipoint services.

F.1.3.1 Multi-netted Server

An example of Multi-netted Server application of asymmetric VLANs is shown in Figure F-4. Note that:

- In the example, the server and both clients are assumed to be VLAN-unaware devices, i.e., they transmit and receive untagged frames only;
- The ingress classification rules assumed by the example are as defined in this standard, i.e., Port-based classification only;
- The configuration shown can only be achieved by management configuration of appropriate values in Static VLAN Registration Entries (8.8.10) in order to configure the indicated member sets and untagged sets.

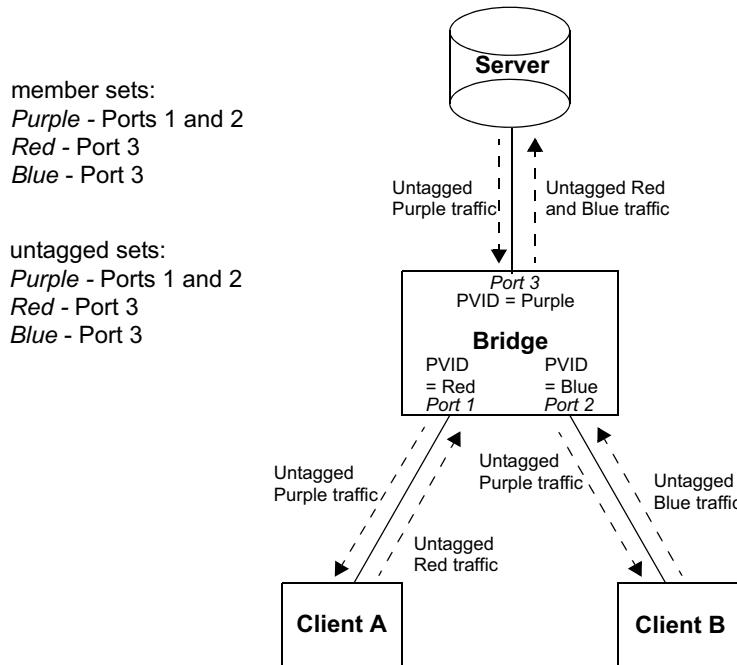


Figure F-4—Asymmetric VID use: “multi-netted server”

In the example, Port-based tagging and an asymmetric VLAN configuration is used in order to permit Clients A and B access to a common server, but to prohibit Clients A and B from talking to each other. Examples of where this type of configuration might be required are if the clients are on distinct IP subnets, or if there is some confidentiality-related need to segregate traffic between the clients.

Client A transmits to the server via Port 1, which will classify this traffic as belonging to VLAN Red; the Bridge, therefore, learns Client A's MAC Address on Port 1 in association with VLAN Red's VID. The Server transmits its responses to Client A via Port 3, which classifies the return traffic as belonging to VLAN Purple. If individual MAC Address learning is configured in the Bridge such that learning is independent between Red and Purple (the VIDs for Red and Purple are allocated to distinct FIDs), then the Bridge will have no knowledge of A in VLAN Purple and will therefore flood the server's responses to Client A on both Port 1 and Port 2. Conversely, if the VIDs for Red and Purple are defined to share the same FID, then the address information learned in Red will be available for use in forwarding the Purple traffic, and the responses to Client A are forwarded only through Port 1.

Similarly, there is a need in this configuration for Blue and Purple to share learning information; hence, in order for this configuration to achieve its objectives, the Red, Blue, and Purple VIDs must be allocated to the same FID in the Bridge.

Hence, in order to construct this particular VLAN configuration, either an SVL Bridge or an SVL/IVL Bridge would be required.

NOTE—The example has been deliberately simplified; in practical applications, the central Bridge would likely be replaced by a number of VLAN-aware Bridges, interconnected with links that would carry the traffic between clients and server as VLAN-tagged frames, with VLAN-tagging and untagging occurring only at the “edge” Ports of the network. An alternative approach to the one described here could also be achieved either by using a VLAN-aware server or by use of more sophisticated ingress classification rules.

F.1.3.2 Rooted-Multipoint

An example of a rooted-multipoint application of asymmetric VLANs is shown in Figure F-5. The defining characteristic of a rooted-multipoint service is that all Bridge Ports that provide access points to the service are designated as either a Root Port or a Leaf Port, and that the Root Ports can communicate with all other Root Ports and all Leaf Ports, whereas the Leaf Ports can communicate with all Root Ports but not with any other Leaf Ports. Services based on rooted-multipoint connectivity are also known as “E-TREE” services as defined in the Metro Ethernet Forum Technical Specification MEF 10.2. Rooted-multipoint connectivity would typically be used by an Internet access provider or a video content provider, where the provider of the service connects to Root Ports, and the subscribers of the service connect to Leaf Ports. This connectivity can be achieved with the “multi-netted server” configuration shown in Figure F-4, but it consumes one VID per subscriber. Figure F-5 shows an alternative that uses just two VIDs: a “Trunk” VID for traffic from Root Ports to Leaf Ports (and to other Root Ports) and a “Branch” VID for traffic from Leaf Ports to Root Ports.

To allow the servers to communicate with all other servers and all subscribers, and to allow the subscribers to communicate with all servers but not with other subscribers, the rooted-multipoint connectivity requires the following configuration:

- a) The server(s) and all subscribers transmit and receive untagged frames only. Note that for a rooted-multipoint service on a Provider Bridged Network, this means the frames do not contain S-TAGs at the customer service interface (a.k.a. the User Network Interface or UNI). Therefore rooted-multipoint can be supported at Port-based service interfaces (15.3) and C-tagged service interfaces (15.4).
- b) The ingress C-VLAN classification may be Port-based or Port-and-Protocol-based for rooted-multipoint on Customer Bridged Networks. The ingress S-VLAN classification may be Port-based or C-TAG-based for rooted-multipoint on Provider Bridged Networks.
- c) In Customer Bridges and S-VLAN Bridges, the indicated values of the member sets and untagged sets (8.8.10) are achieved by management configuration of Static VLAN Registration Entries (8.8.2). In Provider Edge Bridges the Trunk and Branch VIDs are also mapped to the appropriate CNP/PEP pair by management configuration of the Provider Edge Port Table (12.13.3.4, ieee8021PbEdgePortTable 17.6.1.2). Note that connectivity between Leaf Ports is blocked by not including Leaf Ports in the member set associated with the Branch VID.

- d) In Customer Bridges and S-VLAN Bridges the indicated values of the PVID parameters are configured by management. In Provider Edge Bridges the equivalent configuration is achieved by management configuration of the C-VID Registration Table (12.13.3.2, `ieee8021PbCVidRegistrationTable` 17.6.1.2).
- e) The Enable Ingress Filtering parameter (8.6.2) is reset, i.e. ingress VLAN filtering is disabled, on Leaf Ports.
- f) The Trunk and Branch VIDs share the same FID (to enable shared VLAN learning).

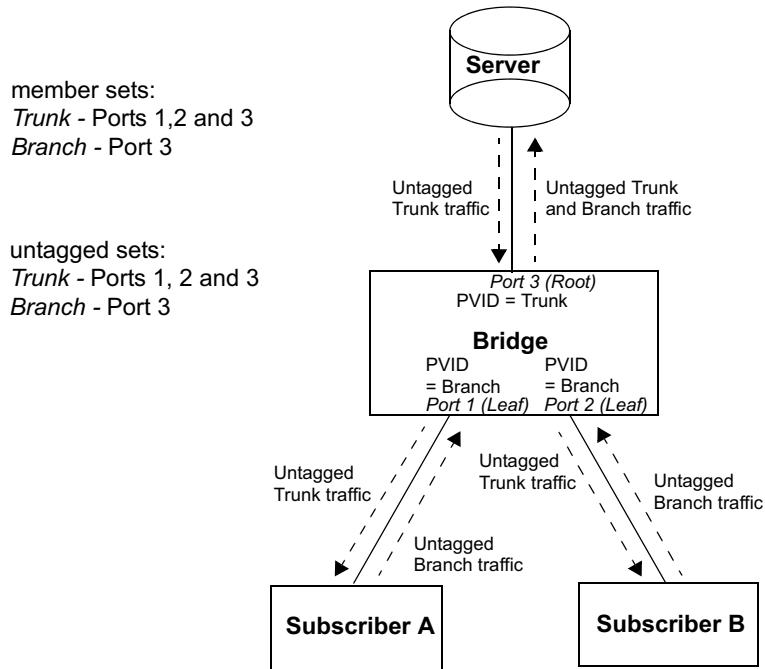


Figure F-5—Asymmetric VLAN use: “Rooted-Multipoint”

NOTE—The example has been deliberately simplified; in practical applications, the central Bridge would likely be replaced by a number of VLAN-aware Bridges, interconnected with links that would carry the traffic between subscribers and servers as VLAN-tagged frames, with only the Root and Leaf Ports of the network carrying untagged frames.

Rooted-multipoint services can be carried on Provider Backbone Bridged Networks by configuring Trunk and Branch S-VIDs as described above, with the Root and Leaf Ports at Customer Network Ports either at an I-component of a Backbone Edge Bridge, or at a Provider Edge Bridge of a PBN which then connects to the PBBN. The I-component of a Backbone Edge Bridge can then either map the Trunk and Branch S-VIDs to separate backbone service instances (i.e. mapped to separate I-SIDs) or bundle them into a single backbone service instance.

Rooted-multipoint connectivity can also be created on a PBBN by configuring Trunk and Branch B-VIDs as described above, with the Root and Leaf Ports at Customer Backbone Ports of a B-component in a Backbone Edge Bridge. One or more backbone service instances can then be mapped to the Trunk and Branch B-VIDs to take advantage of the rooted-multipoint connectivity.

F.1.4 Generic constraints on SVL and IVL use

This subclause describes the general constraints on the mapping of VIDs to FIDs, from the point of view of a given Bridge that learns from or forwards frames on a set of VIDs (the Bridge’s “active set” of VIDs). If

- a) The individual MAC Addresses associated with each point of attachment to the active set of VIDs are unique (i.e., the “Duplicate MAC Address problem” is not present), and
- b) There is no Bridge or Bridge-like device that takes frames from one VID in the active set and subsequently transmits them on another VID in the active set, then

every VID in the active set may share the same FID; in other words, individual MAC Address information learned in the context of any one VID may be used in forwarding decisions taken relative to any of the others, so the SVL approach can be used in that Bridge.

Furthermore, if

- c) Each bidirectional, individual MAC-Addressed, conversation between pairs of end stations makes use of the same VID in both directions, then

every VID in the active set may be allocated a distinct FID (with the possibility of a little extra flooding as learning of addresses in the context of one VID does not contribute to forwarding decisions for that address relative to any other VID). Under these circumstances, rule b) may also be relaxed and restated as follows:

- d) Frames associated with one VID in the active set may be received by (up to) one Bridge and transmitted using another VID in the active set, provided that there is no loop in such VLAN to VLAN forwarding, e.g., for a set of VIDs Red, Blue, and Green, there is no logical loop in copying frames between VIDs, such as copying from Red to Green by one Bridge, Green to Blue by another, and Blue back to Red by a third.

So

- e) If rules a), b), and c) are true, and d) is false, for all VIDs in the active set, then either an SVL or an IVL Bridge can be deployed;
- f) If rules a) and b) are true, and c) and d) are false for all VIDs in the active set, then only an SVL Bridge can be deployed;
- g) If rules a) or b) or d) are false, and c) is true for all VIDs in the active set, then only an IVL Bridge can be deployed.

These conditions are all on the basis that they apply “for all VIDs in the active set.” Clearly, in more complex scenarios, some VIDs in the active set will have requirements that dictate SVL behavior on the part of a given Bridge, while others will have requirements that dictate IVL behavior. Under such circumstances, an SVL/IVL Bridge is required, allowing those VIDs that need to be shared to be mapped to a single FID, while those that need to be independent are mapped to distinct FIDs. Needless to say, wherever an SVL or IVL Bridge can be deployed, it can successfully be replaced by an appropriately configured SVL/IVL Bridge.

F.2 Configuring the Global VLAN Learning Constraints

Subclause F.1 described the requirements that exist for the two approaches to learning in Bridges, closing with some generic rules for how to determine whether, for a given Bridge, SVL, IVL, or SVL/IVL can be successfully deployed. In Bridges, the set of requirements for Independent and/or Shared VLAN Learning is configured as a set of global VLAN Learning Constraint specifications, using the management tools defined in 12.10.3. Two types of VLAN Learning Constraint are defined in 8.8.8.2, which also defines how the set of constraints is used in order to derive a valid mapping of VIDs to FIDs.

The constraint specifications can be constructed on a modular basis. For example, the configuration shown in Figure F-4 has a requirement for Shared VLAN Learning among VIDs Red, Blue, and Purple. This could be expressed as follows:

{Red S Purple};
 {Blue S Purple}

with {Red S Blue} being implied by the transitive nature of the S Constraint.

If we add a similar server access configuration in the same network that requires Red to share with Yellow and Orange, then this could be expressed as

{Red S Yellow};
 {Orange S Yellow}

with {Red S Orange}, {Yellow S Blue}, {Yellow S Purple}, {Orange S Blue}, and {Orange S Purple} being implied by the transitive nature of the S Constraint.

Hence, Red, Blue, Purple, Yellow, and Orange are all required to map to the same FID in order for the set of S Constraints (both explicit and implied) to be met. The constraints that express that requirement are built up from their constituent requirements; namely, for Red and Blue to share with Purple to meet one configuration need, and for Red to share with Yellow and Orange to meet another.

NOTE 1—The five VIDs in this example can be viewed as forming a Shared Set; i.e., a set of VIDs that have a mutual requirement to share learned information—all members of a Shared Set must map to the same FID. Any sequence of S Constraints defines one or more such Shared Sets. Any two Shared Sets can also map to the same FID as long as, for any pair of VIDs, one selected from each Shared Set, no I Constraints require that pair of VIDs to learn independently. Hence, if there are only S Constraints defined, then all VIDs can be mapped to a single FID.

Similarly, the I Constraints can be added on a modular basis. Continuing from the above example, a Bridge-Router (Figure F-1) might be present in the network, which has the effect of connecting VIDs Indigo and Green together, thus creating a requirement for Indigo and Green to be independent. This could be expressed as

{Indigo I 1};
 {Green I 1}

A separate independence requirement might be imposed by the fact that three stations, attached to Indigo, Vermilion, and Red VIDs, all make use of the same individual MAC Address. This could be expressed as:

{Indigo I 2};
 {Vermilion I 2};
 {Red I 2}

Hence, {Indigo, Vermilion, Red} have to be mutually independent (assigned to distinct FIDs), {Indigo, Green} have to be mutually independent, and {Red, Blue, Purple, Yellow, Orange} have to be shared.

The minimum number of FIDs required to satisfy this total constraint specification is three, e.g.:

FID A: Red, Blue, Purple, Yellow, Orange
 FID B: Indigo
 FID C: Green, Vermilion

although an equally valid allocation for three FIDs is

FID A: Green, Red, Blue, Purple, Yellow, Orange
 FID B: Indigo
 FID C: Vermilion

and an equally valid allocation, using the maximum number of FIDs that could be used for this set of constraints and VIDS, is

- FID A: Red, Blue, Purple, Yellow, Orange
- FID B: Indigo
- FID C: Green
- FID D: Vermilion

NOTE 2—It can clearly be seen from this example that it is possible to add further constraints that result in impossible VID to FID mappings; for example, if we were to add {Indigo I Red} or ({Yellow I 3}, {Blue I 3}), then the result is at least one pair of VIDs that have a requirement to both share the same FID and to use distinct FIDs at the same time. Such configurations are examples of Learning Constraint inconsistencies (8.8.8.3).

The assumption behind these constraint specifications is that they are applied globally, in the sense that all Bridges in a given network are configured with the same set of constraints. This assumption is important to ensure that each Bridge is in a position to determine whether, given its current active set of VIDs, it is capable of adopting a VID to FID mapping that will satisfy the specified constraints. If it cannot achieve such a mapping (for any of the reasons identified in 8.8.8.3), then it has detected a network misconfiguration that can only be resolved by management intervention. The managed object specification (12.10.3) provides a Notification for use in such circumstances, to alert a management station to the existence of the problem.

F.3 Interoperability

If the configuration of the network is such that it is not necessary to configure any VLAN Learning Constraints into the Bridges, i.e.:

- a) There are no instances where two (or more) points of attachment to different LANs (and different VIDs) make use of the same individual MAC Address;
- b) There are no instances where a Bridge receives frames with one VID and transmits them with another VID;
- c) There is no asymmetric VLAN use, i.e., there is no pair of end stations for which bidirectional, unicast conversations make use of different VIDs for each direction of transmission,

then it is possible to freely intermix SVL, IVL, and SVL/IVL Bridges in that network, and they can all successfully interoperate.

If the configuration of the network requires one or more S Constraints (and no I Constraints) to be configured into the Bridges, then SVL and SVL/IVL Bridges can be used freely; however, IVL Bridges may only be used in locations where their active set of VIDs does not include any pair of VIDs for which an S Constraint (either explicit or implied) has been defined.

If the configuration of the network requires two or more I Constraints (and no S Constraints) to be configured into the Bridges, then IVL and SVL/IVL Bridges can be used freely; however, SVL Bridges may only be used in locations where their active set of VIDs does not include any pair of VIDs for which I Constraints with the same Independent Set Identifier have been defined.

If the configuration of the network requires both I Constraints and S Constraints to be configured into the Bridges, then SVL/IVL Bridges can be used freely; however,

- d) SVL Bridges can only be used in locations where their active set of VIDs does not include any pair of VIDs for which I Constraints with the same Independent Set Identifier have been defined, and
- e) IVL Bridges can only be used in locations where their active set of VIDs does not include any pair of VIDs for which an S Constraint (either explicit, or implied) has been defined.

Annex G

(informative)

MAC method dependent aspects of VLAN support

G.1 Example tagged IEEE 802.3 EtherType-encoded frame format

Figure G-1 provides an illustrative example of a single tagged EtherType-encoded frame format as used in an Ethernet to Ethernet bridge. This illustration is only of the simplest case, that is, a single-level, fixed-size tagging between identical MACs.⁵²

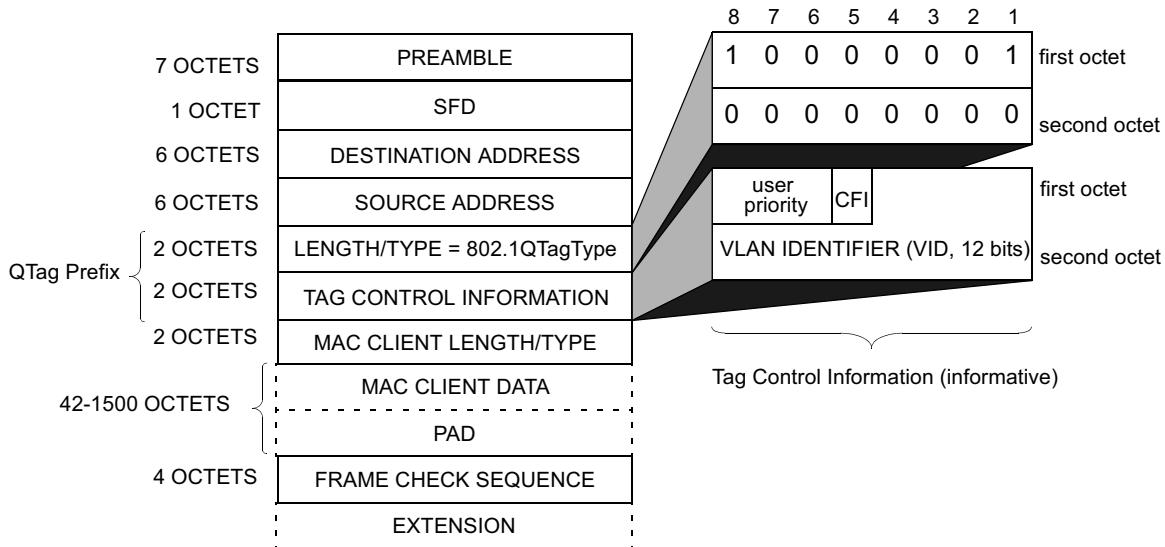


Figure G-1—Example of IEEE 802.3 MAC frame format

G.2 Padding and frame size considerations

G.2.1 Treatment of PAD fields in IEEE 802.3 frames

The minimum frame size constraint placed on IEEE 802.3/Ethernet frames requires frames to carry zero or more pad octets following the MAC client data, in order to ensure that no frame of total length less than 64 octets is transmitted on the medium. This requirement means that frames whose overall length would otherwise be less than 64 octets in length have (64-len) octets of padding added after the MAC client data, where len is the size of the frame before padding.

When tagged frames are transmitted by a Bridge on an IEEE 802.3 MAC, there are two permissible approaches, as follows:

⁵²Figure G-1 was part of Clause 3 of IEEE Std 802.3.

- a) Keep the minimum frame size generated by the Bridge equal to 64 octets. This implies that the number of pad octets in a received untagged IEEE 802.3 frame would be reduced by up to 4 octets when that frame was tagged;
- b) Adopt a minimum tagged frame length of 68 octets. This implies that the number of pad octets in a received untagged IEEE 802.3 frame would not be adjusted when tagging such frames; equally, if subsequently untagged, no pad adjustment would be necessary before transmission on IEEE 802.3/Ethernet.

There is a similar choice to be made in end stations that generate tagged frames:

- c) In some existing implementations, the decision as to whether pad octets are needed will be made at a point where it is impractical to distinguish between tagged and untagged frames. In these cases, the end station will use a minimum frame size of 64 octets for all frames;
- d) In other cases, the padding decision will be taken at a point before it is known whether the frame will be transmitted tagged or untagged. In these cases, the end station will use a minimum tagged frame size of 68 octets and a minimum of 64 octets for untagged frames.

These approaches are all consistent with the IEEE 802.3 frame specification.

The implication is that, for correct operation on IEEE 802.3/Ethernet, all devices have to be capable of correctly handling tagged frames of less than 68 octets in length (G.2.3).

G.2.2 Maximum PDU size

VLAN tagging of an untagged frame, or relaying frames in tagged frame format, can result in an increase in the length of the original frame. If transmission of a given frame in tagged frame format through a given destination Port would result in violation of the maximum PDU size for the destination MAC method, the Bridge discards the frame for that destination Port.

NOTE—Violation of the maximum PDU sizes for destination MAC methods can produce undefined results in networks that contain devices that adhere strictly to these maxima, or in MAC methods where these maxima are inherently constrained by the operation of the MAC method itself (e.g., constrained by timing considerations in the MAC state machines).

IEEE Std 802.3 defines an extension to the normal IEEE 802.3 maximum frame size for the specific purpose of accommodating the additional octets of the VLAN tag header. The example frame translations in this annex make use of this extension to the IEEE 802.3 frame size.

G.2.3 Minimum PDU size

VLAN untagging of a tagged frame results in the original frame decreasing in length.

Where the destination MAC is CSMA/CD:

- a) If untagging a given frame would result in violation of the minimum frame length requirements of CSMA/CD, the Bridge is required to adjust the PAD field to ensure that the frame length equals the minimum length of 64 octets (G.2.1);
- b) If a frame is transmitted in tagged frame format, the Bridge may adopt a minimum tagged frame length of either 64 or 68 octets, as an implementation option. If the latter is chosen, the Bridge adjusts the size of the PAD field on transmission for any tagged frame that is less than 68 octets in length (G.2.1).

Annex H

(informative)

Interoperability considerations

VLAN-aware Bridges that conform to this standard are able to interoperate in networks with other VLAN-aware Bridges. However, the VLAN-based filtering service defined in this standard, as provided in the context of a single spanning tree for the network, involves some constraints on the network topology and individual device configurations that differ from the set of constraints that apply to the building and configuration of networks based only on IEEE Std 802.1D.

In addition, VLAN-aware Bridges are able to interoperate with Bridges conformant with the IEEE 802.1D specification, as well as with both priority-aware and VLAN-aware end systems. Both the VLAN based filtering service and the tag insertion and removal service of IEEE Std 802.1Q cause constraints on intermixed network topologies and device configurations that again differ from the building and configuration of IEEE 802.1D standard networks.

The implications of certain device configurations may not be immediately apparent from the technical detail of this standard. In order to clarify the nature of the additional constraints, H.1 through H.5

- a) Describe the basic requirements for interoperability;
- b) Discuss those requirements in the context of homogeneous and heterogeneous configurations, with examples of some of the problems that can occur if these requirements are not adhered to.

H.1 Requirements for interoperability

Two primary aspects of the configuration of a network are of concern from the point of view of interoperability:

- a) Establishing a consistent view of the static filtering configuration of Bridges in the network;
- b) Ensuring that untagged frames are VLAN-tagged (and that the tag is subsequently removed) consistently regardless of Spanning Tree reconfigurations.

H.1.1 Static filtering requirements

Static filtering controls allow the network administrator to impose a level of control over the permitted connectivity in the network, by setting static MAC Address filters in the Filtering Databases of Bridges, and by controlling the extent of particular VLANs by manipulation of Static VLAN Registration Entries (8.8.2).

In order to ensure that end station-to-end station connectivity (or the lack of it) is consistent in all possible Spanning Tree configurations, any static filters need to be established taking account of the full mesh topology of the physical interconnections between Bridges in the network, not just the “normal” Spanning Tree topology to which the network is configured when all Bridges and LANs are operating correctly. An example of the consequences of failure to establish consistent controls for static VLAN filtering is given in H.2.1.

H.1.2 Configuration requirements for VLAN-tagging

IEEE 802.1Q Bridges classify incoming untagged frames by applying either a Port-based tagging rule on ingress that uses the PVID for the reception Port as the VLAN classification for such frames or a Port-and-Protocol-based rule that uses the frame's upper layer protocol to select one of a port's VIDs. Maintaining consistent connectivity between any pair of end stations that are on the same VLAN, and where one or both of those end stations is VLAN-unaware, requires that

- a) All VLAN-aware Bridge Ports that are connected to the same LAN apply a consistent set of ingress rules (8.6);
- b) All VLAN-aware Bridge Ports that are connected to the same *legacy region* of a network apply a consistent set of ingress rules;
- c) All VLAN-aware Bridge Ports that serve LANs to which members of the same VLAN are (or can be) attached apply a consistent set of ingress rules.

A legacy region of a network consists of any set of LANs that are physically interconnected via VLAN-unaware, IEEE 802.1D Bridges. A legacy region has the property that, by appropriate configuration of the Spanning Tree, a Spanning Tree path could be created between any pair of LANs in the region such that the path would pass only through VLAN-unaware Bridges.

NOTE—In case b), Spanning Tree reconfiguration within the legacy region can change the logical connectivity between the VLAN Ports and the LANs that they (directly or indirectly) serve. Hence, a Spanning Tree reconfiguration could result in any end stations connected to the legacy region being serviced via any VLAN-aware Port. In effect, such a reconfiguration reduces case b) to case a). Figure H-2 and Figure H-3 give examples of this type of configuration. In Figure H-2, the legacy region consists of all three LANs and both IEEE 802.1D Bridges. In Figure H-3, the legacy region consists of the IEEE 802.1D Bridge and both LANs to which it is attached. An example of case c) is where an end station attached to a leaf LAN is in the same VLAN as a server that is attached to a distinct LAN, i.e., all possible Spanning Tree paths between the two stations pass through a VLAN-aware region of the network.

The essence of what these rules express is that if a given untagged frame belongs on a given VLAN, then the classification and tagging behavior of any VLAN-aware Bridges that are required to tag that frame needs to be the same, regardless of the logical connectivity that is created by the Spanning Tree configuration of the network. Examples of the consequences of failure to apply these rules appear in H.3 and H.5.

H.2 Homogenous IEEE 802.1Q networks

This standard requires new considerations in building a network in which all Bridges are VLAN-aware. The arbitrary plug-and-play capability of IEEE Std 802.1D in creating a network topology is restricted when making use of the VLAN extensions defined in this standard.

H.2.1 Consistency of static VLAN filtering

In order for stations that are members of a given VLAN to be able to reach other members of the same VLAN elsewhere in the network, all Ports that are part of the Spanning Tree active topology (i.e., all Ports that are in a forwarding state) connecting the stations must be included in the member set (8.8.10) for the VID that identifies that VLAN. In order for this connectivity to be independent of any reconfiguration of the Spanning Tree topology, all paths among those stations, both forwarding and blocked, must have this characteristic. Use of management controls to manipulate the member set (e.g., filters for security) must be applied in a manner consistent with requirements of the full mesh topology of the network.

An inconsistency occurs, for example, if a VID is restricted from an active path, but not from a redundant path currently blocked by the operation of Spanning Tree. Should a Spanning Tree reconfiguration enable the previously blocked path, the restriction will no longer be in place. In the reverse, a Spanning Tree

reconfiguration may suddenly impose a restriction that had not existed. A common use of such management restriction will likely arise from managers who make use of an “access” port construct. An access port may be a port that is absent from the member set (8.8.10) for all VLANs but the untagged, default VID. Should such an access port become the active connection between two portions of the network as a result of a Spanning Tree reconfiguration, all VLANs but the one identified by the default VID will be partitioned at that point in the topology.

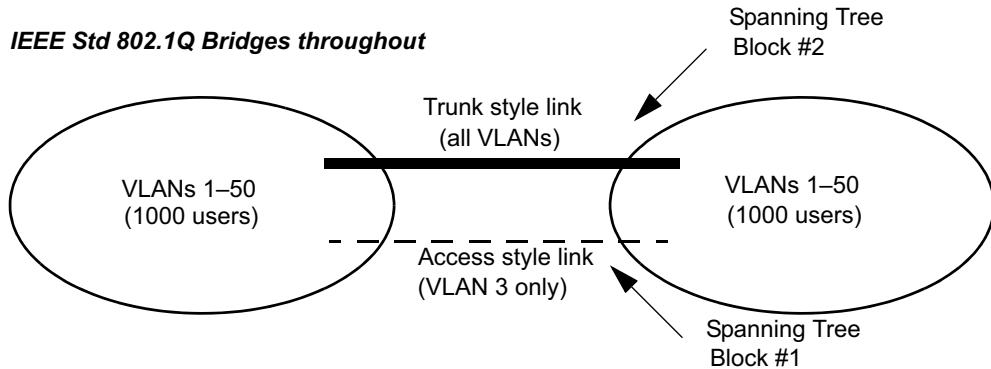


Figure H-1—Static filtering inconsistency

In Figure H-1, the trunk style link and access style link cause a loop through the left and right portions of the network. STP will block one or the other. Should the Spanning Tree block at point #1, all 2000 users may communicate on any of the 50 VLANs. However, should the Spanning Tree block at point #2, the left and right portions of the network will be partitioned on all VLANs excepting VLAN 3 (the VLAN carried via the access style link.)

H.2.2 Consistent view of the “untagged VLAN(s)” on a given LAN

In the Port-based VLAN model defined in this standard, the PVID for a Port provides the VLAN classification for all frames on the attached LAN that are received in untagged form. Any LAN with more than one IEEE 802.1Q Bridge attached has such an “untagged VLAN” for each Bridge. No explicit requirement that PVID be consistent for all Bridges on the same LAN, nor mechanism to assure such, has been included in this standard.

Similarly, in the Port-and-Protocol-based VLAN model defined in this standard, one member of the VID Set for a Port provides the VLAN classification for all frames on the attached LAN that are received in untagged form with a particular upper layer protocol. Any LAN with more than one such IEEE 802.1Q Bridge attached has such a set of “untagged VLANs” for each Bridge. No explicit requirement that these be consistent for all Bridges on the same LAN, nor mechanism to assure such, has been included in this standard.

Consider the case of a LAN to which are attached three VLAN-aware Bridges, each of which is using Port-based classification and is capable of transmission of untagged frames onto the LAN. An untagged frame placed on that LAN by any one of the Bridges will be associated by each of the other two Bridges with its own configured PVID for its reception Port on that LAN. The IEEE 802.1Q VLAN model requires that each frame have a unique VLAN association, and that association is represented by a single, global VID value. Therefore, it follows that all IEEE 802.1Q Bridges on that LAN must make use of the same classification rules (in this case, the same PVID) for their ports connected to that LAN.

It has been suggested that in the special case of a direct point-to-point connection between two IEEE 802.1Q Bridges or other VLAN-aware devices, other rules might apply. No mechanism for identifying such links has yet been suggested.

This creates a configuration challenge for installers of Bridges that conform to this standard. Initial management configuration of the Bridges (the setting of PVIDs, VID Sets, and Protocol Group Databases) must be made consistent among the Bridges, in a manner that takes into account the actual physical topology. Changes to the physical topology may require specific changes to the configuration of all affected switches. These requirements effectively disallow a plug-and-play installation as supported by IEEE 802.1D Bridged Local Area Networks, unless all Bridges are left with their default Port-based classification rules and with each PVID = 1.

H.3 Heterogeneous networks: Intermixing IEEE 802.1D (D) and IEEE 802.1Q (Q) Bridges

This subclause discusses networks in which VLAN-aware Bridges that conform to this standard are intermixed with VLAN-unaware Bridges conformant to IEEE Std 802.1D.

A principal limitation in intermixing Q Bridges with D Bridges is that the VLAN filtering services are not universally available throughout the network. Also, services for the insertion and removal of tags are not universally available. Furthermore, spanning tree reconfigurations may cause filtering services, as well as tag insertion and removal services, to become available or become unavailable independent of actions of affected users.

H.3.1 Example: Adding an IEEE 802.1Q Bridge to provide filtering to an IEEE 802.1D network

Example problems can be shown with the following topology diagrams. Figure H-2 includes one Q Bridge and two D Bridges:

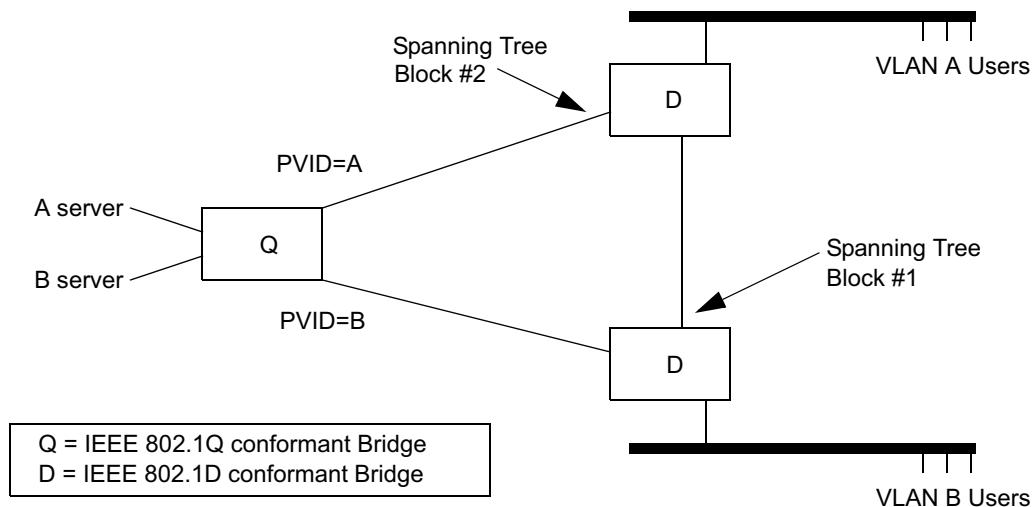


Figure H-2—Interoperability with IEEE 802.1D Bridges: example 1

If the Spanning Tree protocol determines to break the loop among the three Bridges by blocking at point #1, connectivity within each VLAN is as desired. However, should the block occur at point #2, traffic from VLAN A users will pass through both D Bridges and be treated as VLAN B traffic upon arrival in the Q Bridge. Connectivity to the A server will be lost for the A users.

H.3.2 Example: Adding an IEEE 802.1D Bridge to a (previously) Homogenous IEEE 802.1Q Network

A similar problem, demonstrating the impact of placing a D Bridge within an otherwise homogenous Q topology, can be shown by the configuration in Figure H-3. Here we include two Q Bridges and add a single redundant D Bridge:

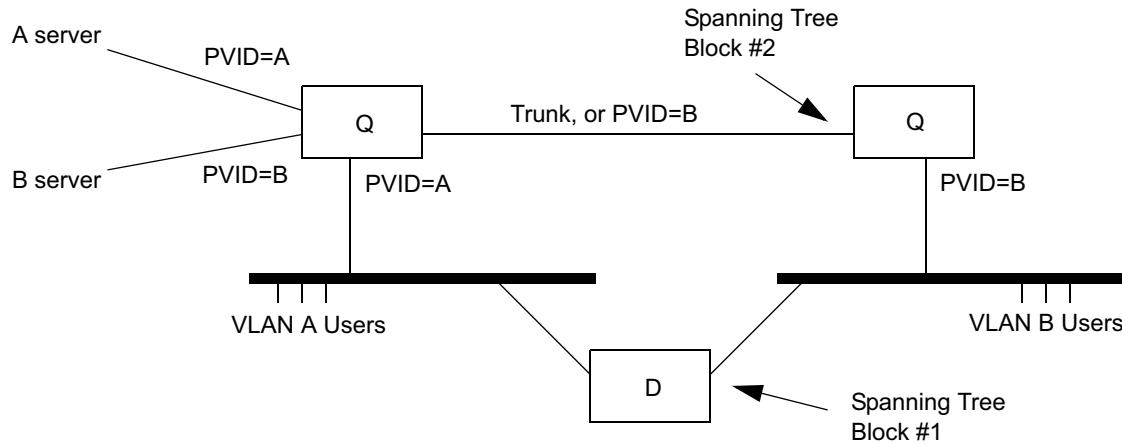


Figure H-3—Interoperability with IEEE 802.1D Bridges: example 2

If STP determines to break the loop among the three Bridges by blocking at point #1, connectivity within each VLAN is as desired. The two Q switches operate as expected. A and B VLAN frames are VLAN-tagged on arrival in either Q Bridge and forwarded only to the appropriate servers. Now suppose an STP reconfiguration results in a block at point #2, but not at #1. This redirects VLAN B user traffic through the IEEE 802.1D Bridge. VLAN B users no longer have their traffic identifiably distinct from VLAN A. An immediate consequence is that the VLAN B users will no longer have access to the “B server.”

H.4 Heterogeneous networks: GARP and MRP issues

H.4.1 GMRP/MMRP: Intermixing IEEE 802.1Q Bridges with IEEE 802.1D Bridges

The specification in this standard for the use of MMRP in VLANs (10.9) makes use of VLAN-tagged frames to signal the MAP Context that applies to the registration information carried in MMRPDUs. IEEE Std 802.1D has not been updated to replace GARP with MRP⁵³; however, such support would necessarily involve the IEEE 802.1D Bridge transmitting MMRP frames untagged, as an IEEE 802.1D Bridge neither generates nor decodes IEEE 802.1Q tag headers. IEEE 802.1D Bridges that implement MMRP as specified in Clause 10 will therefore regard tagged MMRP frames as badly formed, and will discard them on receipt. Using an IEEE 802.1D Bridge to interconnect two or more LAN regions containing IEEE 802.1Q devices that implement MMRP will therefore prevent MMRP attribute propagation between the IEEE 802.1Q

⁵³It is anticipated that this will happen at some point in the future.

regions, with attendant effects upon the forwarding behavior of both the IEEE 802.1D and IEEE 802.1Q Bridges in the LAN. This configuration can be made to work if the IEEE 802.1D Bridge is statically configured with the following:

- a) An All Groups entry in the Filtering Database, specifying Registration Fixed on all Ports, and
- b) The MMRP Protocol Administrative Control parameters set to disable MMRP on all Ports.

As the Bridge no longer supports the MMRP application, it will forward MMRPDUs on all Ports that are in Forwarding. The effect of this is to configure the IEEE 802.1D Bridge to behave in the same manner as an IEEE 802.1D-1993 [B9] Bridge.

Placing IEEE 802.1D Bridges around the periphery of an IEEE 802.1Q-based network works correctly, as long as, for a given IEEE 802.1D Bridge, the IEEE 802.1Q Bridges connected to the same LAN(s) are configured to untag any VLANs that are relevant to the MMRP operation of the IEEE 802.1D Bridge. The IEEE 802.1D Bridge generates untagged MMRP frames, which the IEEE 802.1Q Bridges classify according to the value of the PVID for the reception Port; in a simple configuration of the IEEE 802.1Q Bridges, the Ports that connect to the IEEE 802.1D Bridge are configured for the PVID VLAN to be untagged on egress.

NOTE 1—There may be situations where more complex configurations are required, in which VLANs other than the PVID are configured untagged in order to maintain the correct IEEE 802.1D Bridge filtering behavior.

NOTE 2—For bridges that make VLAN assignments on untagged frames according to Port-and-Protocol-based classification rules, special care is necessary in configuration: since MMRP does not carry information about the particular protocol for which Group membership is intended, it must be taken to apply to all protocols for which untagged traffic is carried on a particular link. Therefore, the VLAN indicated by a port's PVID is used to carry the MMRP frames associated with all of the VIDs that are members of the VID Set of that Port; in addition, the PVID must be configured to egress untagged on any other bridge port where any of the set of VIDs also egresses untagged and requires MMRP operation beyond that egress port.

The effect of this type of configuration is that all registrations propagated by a given IEEE 802.1D Bridge on a given (Port-based or Port-and-Protocol-based) VLAN are seen by all other IEEE 802.1D Bridges served by IEEE 802.1Q Bridges for which the VID for that VLAN is configured for untagged egress. The filtering behavior of the IEEE 802.1D Bridges is therefore governed only by the behavior of other devices (both IEEE 802.1D and IEEE 802.1Q) that are attached to the same VLAN.

H.4.2 Intermixing IEEE 802.1Q Bridges that implement MRP with IEEE 802.1Q Bridges that pre-date MRP

If IEEE 802.1Q Bridges that implement GARP, or were implemented prior to the existence of MRP, are intermixed with IEEE 802.1Q Bridges that implement MRP, then it must be assumed that the non-MRP Bridges will forward or filter MRP frames in accordance with the rules that apply to the group MAC addresses used for propagation of the following MRP frames:

- a) MRP frames with a destination address defined in Table 10-1 will be forwarded by pre-MRP bridges.
- b) MRP frames with a destination address defined in Table 8-1 will be filtered by all bridges.
- c) MRP frames with any other destination address will be forwarded or filtered in accordance with the state of the filtering database in the bridge concerned.

In the case of MVRP information propagation in Provider Bridges, this leads to some consequences:

- d) Any Customer Bridge that is placed within a network of Provider Bridges will not forward MVRP frames generated by the Provider Bridges; propagation of MVRP information through the network of Provider Bridges will therefore be incorrect.

- e) Any Provider Bridge that does not implement MRP that is placed within a network of Provider Bridges will forward MVRP frames as ordinary multicast data, but will not affect the information contained in those frames; propagation of MVRP information through the network of Provider Bridges may therefore be incorrect.

H.5 Intermixing Port-based classification and Port-and-Protocol-based classification or future enhancements in IEEE Std 802.1Q

The discussion in H.4 on intermixing Q Bridges with D Bridges has a direct analogue in the mixing of bridges implementing only Port-based and Port-and-Protocol-based classification of frames in IEEE 802.1Q networks. This revision and potential subsequent editions of IEEE Std 802.1Q extend the VLAN classification capabilities to support more sophisticated ingress rules for frame classification.

In VLAN configurations that use both Port-based and Port-and-Protocol-based VLAN classification, a Bridge that supports only Port-based VLAN classification will merge VLANs that would otherwise be classified separately by a Bridge that supports Port-and-Protocol-based VLAN classification. To get around this problem, it may be possible to dedicate specific Ports to specific protocols in Bridges that support only Port-based VLAN classification, as in the example shown in Figure H-4. However, such solutions may not be possible where there are multiprotocol end stations in the network.

H.5.1 Example: Intermixing Protocol-based ingress rules

Consider the case where Bridges implement a configuration mechanism to select between Port-based classification rules (a “Q-Port Bridge”) and Port-and-Protocol-based rules (a “Q-Port/Protocol Bridge”). This case would allow, for example, for support for IP and IPX as distinct VLANs. The topology shown in Figure H-4 might apply when a Q-Port/Protocol Bridge is added to a topology; otherwise, using only Q-Port bridges allows users of two protocols to participate in two separate VLANs.

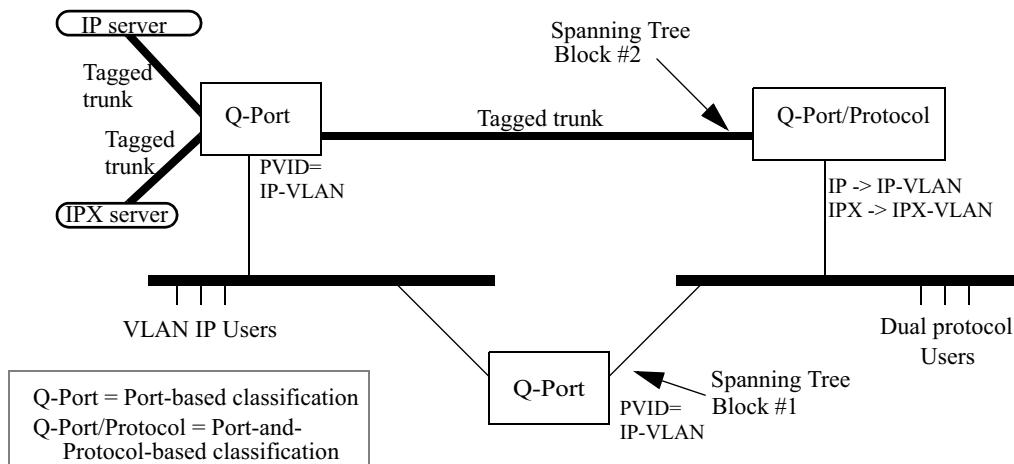


Figure H-4—Interoperability between Port-based and Port-and-Protocol-based classification

Consider this network, when STP has blocked at point #1, and not at #2. The upper Q-Port Bridge operates as expected, and the Q-Port/Protocol Bridge provides Port-and-Protocol-based classification for the frames received from the dual protocol users on the right-hand LAN. IP-VLAN and IPX-VLAN frames are VLAN-tagged on the trunks to and from the uppermost Bridges and servers. But if a STP reconfiguration should result in a block at point #2, but not at #1, activating traffic through the lower Q-Port Bridge, dual protocol

users will have all their traffic treated as part of the IP-VLAN. An immediate consequence is that the uppermost Bridge will no longer provide them access to the “IPX server.” It is the fact that the lower Q-Port Bridge must have just one of the two VLANs configured for all untagged traffic, regardless of its protocol, that leads to this lack of connectivity.

H.5.2 Differing views of untagged traffic on a given LAN

Further challenges arise when one considers the case where several Q Bridges, some implementing Port-based and some implementing Port-and-Protocol-based classification, all attach to the same LAN. Again, the rule that any given frame exists in exactly one VLAN requires that all of these Bridges be configured with consistent ingress rules. In this case, the Q-Port Bridges will provide a least common capability, and this further requires common configuration of the PVID in the Q-Port and Q-Port/Protocol Bridges.

Annex I

(informative)

Priority and drop precedence

This standard allows management of priorities, flow metering, queue assignment, and queue servicing. This annex documents the rationale for the recommended and default priority to traffic class mappings in Table 8-4 and the encoding of priority and drop eligibility in Table 6-3 and Table 6-4.

Classification of user data frames into a small number of behavior aggregates, together with aggregate dependent forwarding behavior in each Bridge, allows signaling of application requirements to the network. Frame classification, aggregate bandwidth metering, policing, and drop precedence marking also facilitate network scaling and provision of services to independent customers through allocation of those functions to appropriate Bridges in the network.

While there are many possible ways to classify frames and to specify forwarding behaviors, it is widely appreciated that a set of well-known and easily understood defaults can facilitate interoperability and the deployment of services. The defaults described in this annex and supported by this standard were chosen to support integrated and differentiated services, to minimize the burden of management, to reduce the possibility of misconfiguration and out-of-order frame delivery, and to provide useful service without management in many networks.

This standard mandates support for strict priority frame transmission (8.6.6) but permits the use of additional traffic class-based transmission selection algorithms. The default assignments of frames to traffic classes on the basis of frame priority, as described in this annex, also support the use of frame priority to select general traffic class-based forwarding behavior.

NOTE—Annex M provides references to the IETF work on integrated and differentiated services ([B14] through [B18] and [B21] through [B30]).

I.1 Traffic types

A full description of the QoS needs of applications and network services is too complex to be represented by a simple number 0 through 7. The pragmatic aim of traffic classification is to simplify requirements to preserve the high-speed, low-cost characteristics of Bridges. At the margin, potential bandwidth efficiency is traded for simplicity and higher speed operation—historically a good decision in the LAN.

The following list of traffic types, each of which can benefit from simple segregation from the others, are of general interest:

- a) Network Control—characterized by a guaranteed delivery requirement to support configuration and maintenance of the network infrastructure.
- b) Internetwork Control—in large networks comprising separate administrative domains there is typically a requirement to distinguish traffic supporting the network as a concatenation of those domains from the Network Control of the immediate domain.
- c) Voice—characterized by less than 10 ms delay and, hence, maximum jitter (one way transmission through the LAN infrastructure of a single campus).
- d) Video—characterized by less than 100 ms delay, or other applications with low latency as the primary QoS requirement.
- e) Critical Applications—characterized by having a guaranteed minimum bandwidth as their primary QoS requirement and subject to some form of admission control to ensure that one system or

application does not consume bandwidth at the expense of others. The admission control mechanism can range from preplanning of the network requirement at one extreme to bandwidth reservation per flow at the time the flow is started at the other.

- f) Excellent Effort—or “CEO’s best effort,” the best-effort type services that an information services organization would deliver to its most important customers.
- g) Best Effort—for default use by unprioritized applications with fairness only regulated by the effects of TCP’s dynamic windowing and retransmission strategy.
- h) Background—bulk transfers and other activities that are permitted on the network but that should not impact the use of the network by other users and applications.

I.2 Managing latency and throughput

Use of priorities and queuing by traffic classes, each class encompassing one or more priorities, facilitates improvement and management of latency and throughput, allowing QoS goals to be supported at higher levels of network loading than would otherwise be possible.

Congestion, resulting in QoS degradation, is not equally likely at all Bridges in a network. Transient traffic patterns are likely to result in congestion in only a few Bridges at a time, while over an extended period, momentary congestion is more likely to occur in the network core than at Bridge Ports attached to one or a relatively small number of end stations. Use of fewer traffic classes for those Ports can lower the cost of implementation and management, and this standard facilitates the use of Bridges supporting differing numbers of classes within a single network that delivers a consistent set of QoS parameters for each frame priority level. Although the number of traffic classes supported by each Bridge Port along the path taken by a given flow of data can vary, the default mappings of priorities to classes ensures that frame ordering is preserved as required by 8.6.6.

With few classes, the focus is on meeting latency requirements—the bandwidth surplus required in a bursty data environment to guarantee sub-10 ms delays without a distinct traffic classification is uneconomically large. As the number of traffic classes that can be used increases, the focus shifts to managing throughput.

The simple default queue servicing policy defined in this standard, strict priority, supports latency management. Active management of bandwidth sharing necessarily requires some management.

I.3 Traffic type to traffic class mapping

Table I-1 groups the traffic types introduced to match the number of traffic class queues supported by a Bridge Port. Each grouping of types is shown as *{Distinguishing type, Type, Type, ...}*. The “distinguishing type” is not treated in any way differently in a Bridge but is italicized here to illustrate, for any given number of queues, which traffic types have driven the allocation of types to classes.

Table I-1—Traffic type to traffic class mapping

Number of queues	Traffic types
1	{ <i>Best Effort</i> , <i>Background</i> , <i>Excellent effort</i> , <i>Critical Applications</i> , <i>Voice</i> , <i>Video</i> , <i>Internetwork Control</i> , <i>Network Control</i> }
2	{ <i>Best Effort</i> , <i>Background</i> , <i>Excellent effort</i> , <i>Critical Applications</i> } { <i>Voice</i> , <i>Video</i> , <i>Internetwork Control</i> , <i>Network Control</i> }
3	{ <i>Best Effort</i> , <i>Background</i> , <i>Excellent effort</i> , <i>Critical Applications</i> } { <i>Voice</i> , <i>Video</i> } { <i>Network Control</i> , <i>Internetwork Control</i> }
4	{ <i>Best Effort</i> , <i>Background</i> } { <i>Critical Applications</i> , <i>Excellent effort</i> } { <i>Voice</i> , <i>Video</i> } { <i>Network Control</i> , <i>Internetwork Control</i> }
5	{ <i>Best Effort</i> , <i>Background</i> } { <i>Critical Applications</i> , <i>Excellent effort</i> } { <i>Voice</i> , <i>Video</i> } { <i>Internetwork Control</i> } { <i>Network Control</i> }
6	{ <i>Background</i> } { <i>Best Effort</i> } { <i>Critical Applications</i> , <i>Excellent effort</i> } { <i>Voice</i> , <i>Video</i> } { <i>Internetwork Control</i> } { <i>Network Control</i> }
7	{ <i>Background</i> } { <i>Best Effort</i> } { <i>Excellent effort</i> } { <i>Critical Applications</i> } { <i>Voice</i> , <i>Video</i> } { <i>Internetwork Control</i> } { <i>Network Control</i> }
8	{ <i>Background</i> } { <i>Best Effort</i> } { <i>Excellent effort</i> } { <i>Critical Applications</i> } { <i>Video</i> } { <i>Voice</i> } { <i>Internetwork Control</i> } { <i>Network Control</i> }

The step-by-step breaking out of traffic types as more classes are available proceeds as follows:

- a) With a single queue, there are no choices. All traffic is Best Effort.
- b) To support integrated services in the presence of bursty best effort data, it is necessary to segregate all time-critical traffic. The amount of high-priority traffic will be restricted by the need to support low latency for Voice, which becomes the defining type for the additional queue.
- c) Two queues may be adequate for Bridge Ports attaching to end stations. The stability of the network as a whole may be unaffected by the performance of configuration protocols on those Ports, and in-band management of the Bridge is likely to occur through another Port. For Bridges within the network infrastructure, a further queue is used to isolate Network Control from the user data traffic.
- d) Traffic for business Critical Applications is separated from Best Effort to allow a bandwidth guarantee to be provided.
- e) The queue separation so far provided can support a large network. The next queue is allocated to distinguishing Internetwork Control traffic from local Network Control.
- f) Background is separated from Best Effort to minimize the effect of bulk transfers on ordinary network use.

- g) Excellent Effort is separated from Critical Applications, either to provide a simple superior service based on policy controlled access or to provide an additional segregated bandwidth guarantee.
- h) The final provides increased network utilization as the higher bandwidth traffic associated with Video is no longer given the same latency guarantee as Voice.

This description is illustrative rather than definitive of the logic of allocating traffic types to classes. The mappings in Table 8-4 support the assignment of other semantics to each traffic type identified by priority values, e.g., the identification of all three illustrative types “Video,” “Critical Applications,” and “Excellent Effort” with assured forwarding classes that provide segregated bandwidth guarantees. However, alternate semantics should take into account the service provided by Bridges with limited traffic class queuing; e.g., of the foregoing, only “Video” would receive priority treatment by default at a Bridge Port supporting queuing for only two classes.

I.4 Traffic types and priority values

Table I-2 shows the correspondence between traffic types and priority values used to select the defaults in Table 8-4. The default priority used for transmission by end stations is 0. Changing this default would result in confusion and likely in interoperability problems. At the same time, the default traffic type is definitely Best Effort. 0 is thus used both for default priority and for Best Effort, and Background is associated with a priority value of 1. This means that the value 1 effectively communicates a lower priority than 0.

Table I-2—Traffic type acronyms

Priority	Acronym	Traffic type
1	BK	Background
0 (Default)	BE	Best Effort
2	EE	Excellent Effort
3	CA	Critical Applications
4	VI	“Video,” < 100 ms latency and jitter
5	VO	“Voice,” < 10 ms latency and jitter
6	IC	Internetwork Control
7	NC	Network Control

Table I-3 summarizes Table I-1, showing just the defining traffic types. By maintaining the groupings of types established for a given number of queues for all less numbers, the table preserves the order of frames of any given type, independent of the number of queues.

This discussion of traffic types, and the suggested association of each with a priority value, differs from the similar discussion in Annex G of IEEE Std 802.1D-2004 and prior revisions of that standard. The latter was developed contemporaneously with IETF Intserv and predates Diffserv. The discussion in this annex better aligns with current practice; in particular, Voice is associated with priority 5, matching the setting of the relevant bits for Expedited Forwarding (EF) in the DSCP (Differentiated Services Code Point) for IP and in the common use of the EXP bits for MPLS. Standards for DSCPs are believed to be the prime reference for use of priority by end stations, and there is no direct change to the behavior of Bridge implementations conforming to this standard as a result of this change.

Table I-3—Defining traffic types

Number of queues	Defining traffic type							
1	BE							
2	VO			BE				
3	NC		VO		BE			
4	NC		VO		CA		BE	
5	NC	IC	VO		CA		BE	
6	NC	IC	VO		CA		BE	BK
7	NC	IC	VO		CA	EE	BE	BK
8	NC	IC	VO	VI	CA	EE	BE	BK

The priority to traffic class mappings in Table 8-4 differ in one minor respect from those specified in prior revisions of this standard and in IEEE Std 802.1D-2004 and its prior revisions. Priority value 2 was previously described as ‘Spare’ and positioned lower than 0 (Best Effort) in priority order. This change may result in networks, including bridges, conformant to prior revisions of this standard, and implementing four or more traffic classes, providing less-than-expected priority to traffic described in this annex as Excellent Effort, and misordering drop eligible traffic for Critical Applications. The change allows better use of the available traffic classes, given the low demand for two distinct priorities of lesser importance than Best Effort, and the best use of PCP when encoding drop eligibility.

I.5 Supporting the credit-based shaper algorithm

Table 8-4 defines a set of recommended priority to traffic class mappings where the credit-based shaper algorithm (8.6.8.1) is supported by one or two of the available traffic classes; the recommended mappings shown are intended for use where priority 5 is used to support SR class A and priority 4 is used to support SR class B.

In order for the credit-based shaper algorithm to operate in the way it is intended, it is necessary to ensure that the shaper algorithm is supported on the numerically highest traffic class(es). Hence, if two traffic classes are used to support the shaper algorithm, it follows that the minimum useful number of traffic classes that a Port could support is three.

Table I-4 redraws Table I-3 for the case where SR class A is supported. The defining traffic type for the numerically highest traffic class is VO in all cases; for 3 through 8 traffic classes, the remaining traffic types are broken out in the same order as in Table G-3.

Table I-5 redraws Table I-3 for the case where SR classes A and B are supported. The defining traffic types for the two numerically highest traffic classes are VO and VI in all cases; for 4 through 8 traffic classes, the remaining traffic types are broken out in the same order as in Table I-3.

Table I-4—Defining traffic types—Credit-based shaper support of one SR class

Number of queues	Defining traffic type							
2	VO	BE				BE		
3	VO	NC			BE			
4	VO	NC			CA		BE	
5	VO	NC	IC		CA		BE	
6	VO	NC	IC		CA		BE	BK
7	VO	NC	IC		CA	EE	BE	BK
8	VO	NC	IC	VI	CA	EE	BE	BK

Table I-5—Defining traffic types—Credit-based shaper support of two SR classes

Number of queues	Defining traffic type							
3	VO	VI	BE				BE	
4	VO	VI	NC		BE			
5	VO	VI	NC		CA		BE	
6	VO	VI	NC	IC	CA		BE	
7	VO	VI	NC	IC	CA		BE	BK
8	VO	VI	NC	IC	CA	EE	BE	BK

I.6 Supporting drop precedence

It is often desirable to meter traffic from different users to ensure fairness or to meet bandwidth guarantees; however, dropping all traffic in excess of a committed rate is likely to result in severe under-utilization of the networks, because most traffic sources are bursty in nature. At the same time, it is burdensome to meter traffic at all points in the network where bandwidth contention occurs. One solution is to mark those frames in excess of the committed rate as drop eligible on admission to the network.

This standard allows drop eligibility to be conveyed separately from priority in VLAN tags so that all previously introduced traffic types can be marked as drop eligible. To provide compatibility with previous revisions of this standard while allowing drop eligibility to be conveyed in Customer VLAN tags (C-TAGs), this standard also allows a subset of the priorities to be conveyed along with drop eligibility marking for some of those priorities within the Priority Code Point (PCP) field of both S-TAGs and C-TAGs.

I.7 Priority code point allocation

The Priority Code Point of VLAN tags allows encoding of five, six, seven, or eight distinct priorities, with a single level of drop eligibility on three, two, one, or zero of those priorities, respectively. Table 6-3 and Table 6-4 specify encoding and decoding for five through eight priorities. The tables are consistent with the following step-by-step reduction in the number of distinct priorities to provide drop eligibility for certain traffic types:

- a) If eight distinct priorities are required, drop eligibility cannot be encoded in the PCP (but may be encoded in VLAN tags using the DEI).
- b) Drop eligibility in support of QoS maintenance for traffic conforming to a committed rate is most effective when used to support time-critical traffic. If seven priorities, one of which can be marked as drop eligible, are required, then the traffic class queuing distinction between Voice and Video is sacrificed to providing drop eligibility for the combined traffic types. This does not preclude marking all Video traffic as drop eligible upon ingress to a network, so as to provide the same guarantee to Voice as a distinct priority.
- c) The distinctions between Critical Applications and Excellent Effort, and between Best Effort and Background traffic types, is removed to provide drop eligibility for Critical Applications and for Best Effort.
- d) Although the use of four priorities, each with drop eligibility, is possible, it is not recommended. Combining Network Control with Internetwork Control could only serve to increase the guarantees provided to the latter at the expense of the former, which if not delivered threatens the stability of the overall network in any case. Moreover, both traffic types should be supportable with guaranteed bandwidth if the network is to be operated successfully.

Choosing first to combine Video and Voice, and then Critical Applications with Excellent Effort (for six queues), provides consistency with the allocation of priorities to traffic classes in the absence of drop eligibility. Bridges that do not implement drop eligibility, but are configured to use the same number or fewer traffic classes, will not misorder frames. If such a bridge is configured to use only five traffic classes, and in accordance with Table 8-4, it will not misorder frames with a priority code point encoded using any of the alternatives provided by Table 6-3.

I.8 Interoperability

Encoding of drop eligibility within the PCP, as opposed to explicitly with the DEI, provides interoperability with bridges that are not capable of supporting the DEI. It also provides compatibility with the use of the MPLS EXP bits to convey priority and drop eligibility.

However, the requirement to provide different combinations of priorities with drop eligibility within the confines of the PCP means that priority and drop eligibility information can be lost for frames traversing a network if the combinations used on individual LANs differ. Use of the DEI does not suffer from this problem. In some cases, loss of drop eligibility information at the boundary between administrative domains risks impacting profile conformant traffic from some users with out-of-profile drop eligible traffic from others; in other cases, the drop eligible marking has already done its job (the next LAN, for example, might deliver traffic to a single customer). Without explicit management, a Bridge Port cannot decide, so the Require Drop Encoding parameter (8.6.7) has been provided.

If bridges attached to the same LAN encode and decode the PCP differently, then incorrect priority values can be attributed and subsequent misordering of frames can occur. Misordering will not occur, with the recommended priority to traffic class mappings of Table 8-4 and the recommended PCP encoding and decoding in Table 6-3 and Table 6-4, if the Bridge performing the incorrect decoding assumes fewer priorities than are actually encoded or if all bridges subsequently transited by the frame use the same number or fewer traffic classes than those used for the encoding. However, incorrect decoding will in all probability

affect other service guarantees that the network is intended to support. If a Bridge can be used in a network that encodes drop eligibility in the PCP, and there is any likelihood of the Bridge being brought into service prior to the network dependent service level configuration, then five priorities, three with drop eligibility (5P3D encoding and decoding), should be used. Bridges that do not support drop precedence should be configured to support five or fewer traffic classes in the same circumstances.

The use of separate Priority Code Point Encoding and Priority Code Point Decoding Tables for each Bridge Port allows adaptation between the encoding scheme in one domain of the network and the encoding scheme used in another to be accomplished in only one of any pair of bridges, each serving as a boundary of its domain, connected by a point-to-point LAN. However, if more than two bridges are attached to a LAN, all need to use the same encoding so that each of its recipients can assign the correct priority to the frame.

The default PCP encoding and decoding, as documented in Table 6-3 and Table 6-4, are reproduced in Table I-6 and Table I-7, with the addition of the default allocation of priorities to traffic classes to the latter.

NOTE—The sequence of the columns for the priority values of 0 and 1 in Table I-6 and Table I-7 are reversed from the sequence in Table 6-3 and Table 6-4 in order to show the alignment with the traffic classes in Table I-3.

Table I-6—Priority Code Point encoding

priority drop_eligible		7	7DE	6	6DE	5	5DE	4	4DE	3	3DE	2	2DE	0	0DE	1	1DE
PCP	8P0D	7	7	6	6	5	5	4	4	3	3	2	2	0	0	1	1
	7P1D	7	7	6	6	5	4	5	4	3	3	2	2	0	0	1	1
	6P2D	7	7	6	6	5	4	5	4	3	2	3	2	0	0	1	1
	5P3D	7	7	6	6	5	4	5	4	3	2	3	2	1	0	1	0

Table I-7—Priority Code Point decoding

PCP		7	6	5	4	3	2	0	1
priority drop eligible	8P0D	7	6	5	4	3	2	0	1
	7P1D	7	6	4	4DE	3	2	0	1
	6P2D	7	6	4	4DE	2	2DE	0	1
	5P3D	7	6	4	4DE	2	2DE	0DE	0
	1	BE							
number of traffic classes	2	VO				BE			
	3	NC		VO		BE			
	4	NC		VO		CA		BE	
	5	NC	IC	VO		CA		BE	
	6	NC	IC	VO		CA		BE	BK
	7	NC	IC	VO		CA	EE	BE	BK
	8	NC	IC	VO	VI	CA	EE	BE	BK

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) [B5], as defined by the Metro Ethernet Forum.
- 3) Underlying layers of OAM, such as MPLS OAM or ATM OAM,⁵⁴ 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 Y.1731 (02/2008).
- 5) Network performance measurement, being defined by the Metro Ethernet Forum and ITU-T, including ITU-T Y.1731.

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 Y.1731. 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 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 MAC Address 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.

⁵⁴For references to these technologies, see the bibliography in Annex M.

- 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 802.1Q CFM to other standards

ITU-T Y.1731 (02/2008) 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.

Table J-2—IEEE / ITU-T terminology differences

IEEE Std 802.1Q	ITU-T Y.1731 (02/2008)
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)

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 (20.28.1) 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.

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 (20.41.2.2). 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 (20.41.2.4) 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 (20.41.2.3) in any LTR entry is unique only over those LTR entries with matching values of the ltrReplyTTL variable.

- d) The ltrLastEgressId variable (20.41.2.3) 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.⁵⁵ 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 13.11), 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.

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,

⁵⁵Bridge 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 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 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.

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 20-16, 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 20.53, 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 20-16 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.

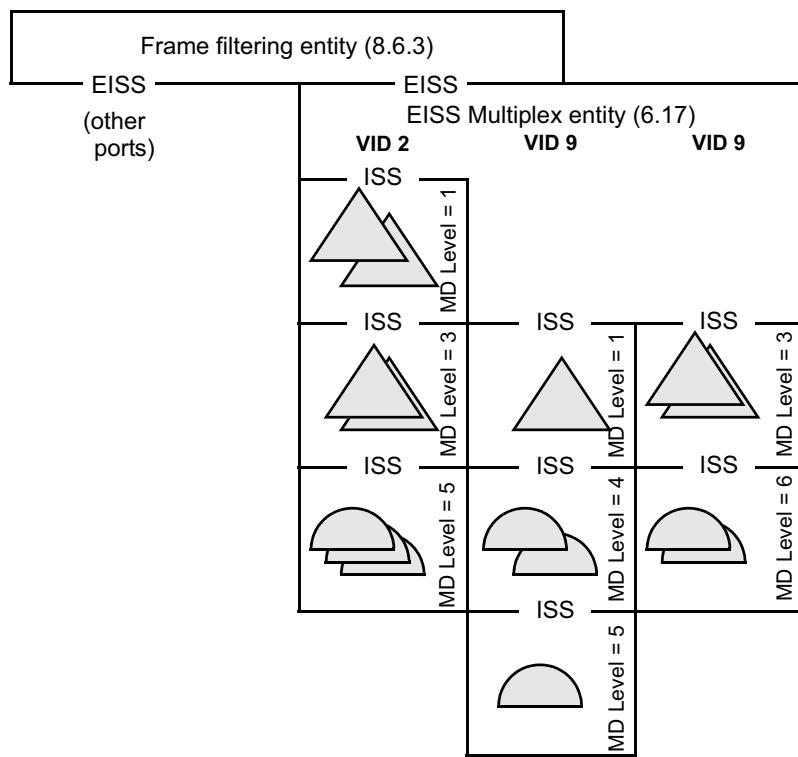


Figure J-1—Up MPs in a CFM Port

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 (20.50) and a single MEP Loopback Initiator transmit state machine (20.32), 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 (20.32) and a single MEP Loopback Initiator receive state machine (20.34), 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.

Annex K

(informative)

TPMR use cases

This material is intended to assist the reader of this standard in putting the operation of a TPMR in context with the operation of Bridged LANs as a whole, by illustrating some of the uses of a TPMR in conjunction with other types of bridge. The set of cases is not, and is not intended to be, exhaustive in terms of all of the possible usage scenarios.

K.1 Use case 1—TPMR as User to Network Interface (UNI) demarcation device

Figure K-1 illustrates a use case where the TPMR performs the function of a UNI demarcation device between an customer network and a provider network. In this illustration, the user link between the customer bridge and the TPMR, and the extension link between the TPMR and the provider Bridge, are both assumed to be IEEE 802.3 links, although, with the exception of the use of IEEE 802.3 EFM OAM on the extension link, these links could in principle be provided by use of any IEEE 802 LAN technology.

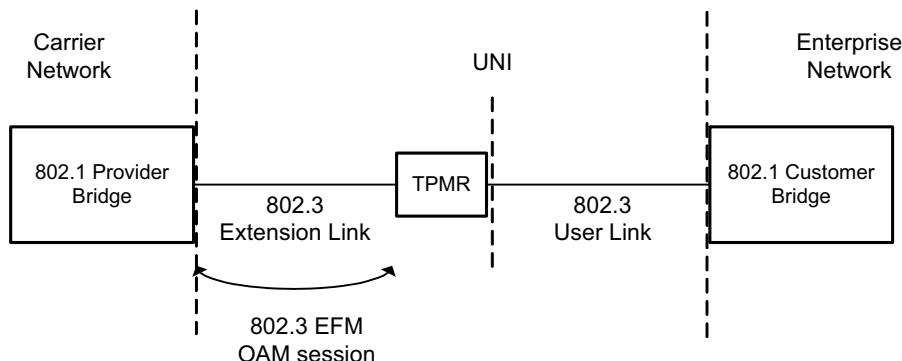


Figure K-1—TPMR as UNI demarcation device

The use of EFM OAM on the extension link allows the service provider to

- Put the TPMR into an intrusive loopback for out-of-service testing.
- Query the IEEE 802.3 MIB in the TPMR.
- Be notified of critical events (Link Fault, Dying Gasp, Critical Event).
- Be notified of link performance events (Errored Symbol Period event, Errored Frame event, Errored Frame Period event, Errored Frame Seconds Summary event).

The potential exists with EFM OAM (on the extension link) to make use of extensions, via organization specific OAMPDUs, to manage the Port of the TPMR that faces the user link.

Alternatively, the TPMR might be managed using SNMP over UDP/IP either in-band on the IEEE 802.3 extension link or out-of-band through separate management connector on the TPMR.

K.2 Use case 2—TPMRs with aggregated links

Figure K-2 illustrates a use case where two TPMRs are used within paired extension/user links that are aggregated at their end points (the Bridges). The TPMRs are transparent to the operation of the IEEE 802.1AX Link Aggregation Control Protocol (LACP), because the Provider Bridge and Customer Bridge are configured to use the “Nearest non-TPMR Bridge group address” as the destination address for LACP; from the point of view of link aggregation, each paired extension link and user link, with the TPMR joining them together, appears to be a single IEEE 802.3 LAN.

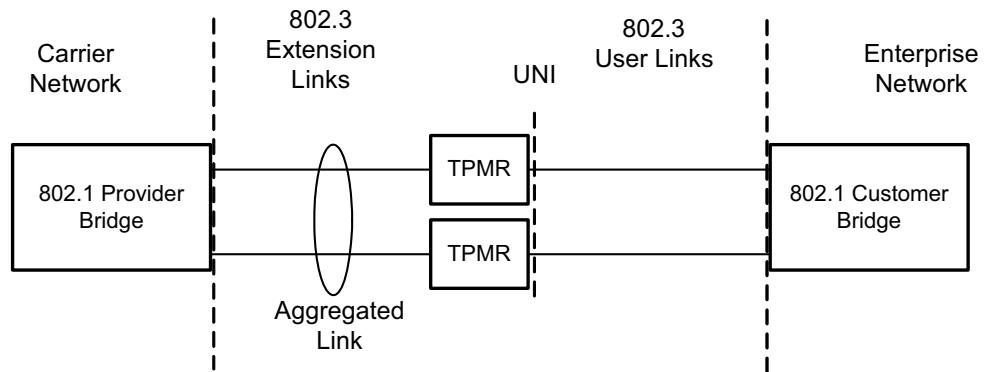


Figure K-2—TPMRs with aggregated links

The TPMRs would be managed separately, via EFM OAM, as in use case 1 (K.1). It should be noted that the use of SNMP to manage the TPMRs in this scenario is problematic, as the aggregated link appears to the higher layers to be a single link, but individual conversations are allocated by the frame distributor in the aggregator to the links in the aggregation in a nondeterministic manner.

K.3 Use case 3—Multiple TPMRs

Figure K-3 shows a variant of use case 1 (K.1) where two extension links and a user link are used, interconnected using two TPMRs.

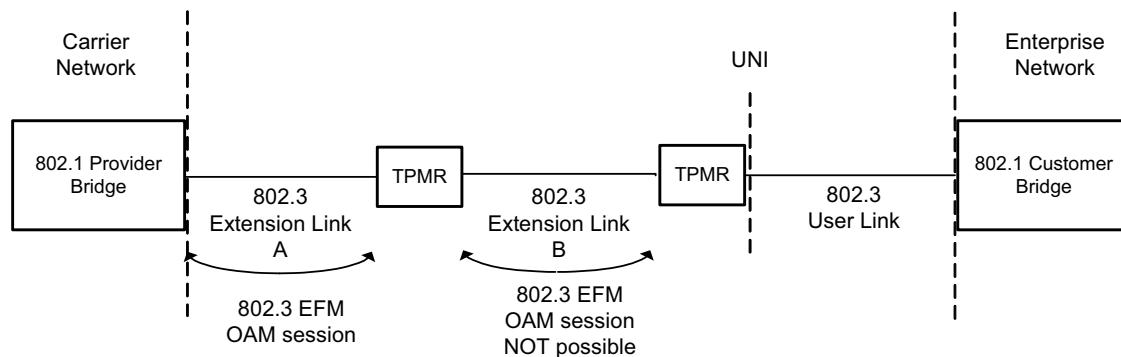


Figure K-3—Multiple TPMRs

Some commercial TPMRs used as demarcation devices power up in EFM OAM Passive mode, waiting to hear from the Active mode master device at the edge of the provider network. A passive-to-passive EFM OAM session, as would occur on extension link B, would never activate, making it impossible to manage the customer-facing TPMR by this means, unless significant changes were made to the capabilities of EFM

OAM to allow some kind of relay capability. On the other hand, by using SNMP as the management protocol, this problem would be avoided.

K.4 Special cases

Figure K-4 (new connectivity) and Figure K-6 (connectivity failure) represent the target cases for the MAC status propagation design.

Figure K-4 shows recovery of the individual LAN at one end of a TPMR chain. The result is to “blip” the MAC_Operational status at the other end of the chain, delaying availability of the recovered link by slightly more than $T_w + T_d$. The figure also shows a possible effect of timing relationships between T_r and T_d , as the initiator of the change retries its Add message transmission just as the TPMR at the other end of the chain returns a confirm. The crossing of messages shown is unlikely, as the time sequence diagrams overemphasize the delay experienced by the Add and Confirm messages, but the effect is benign in any case.

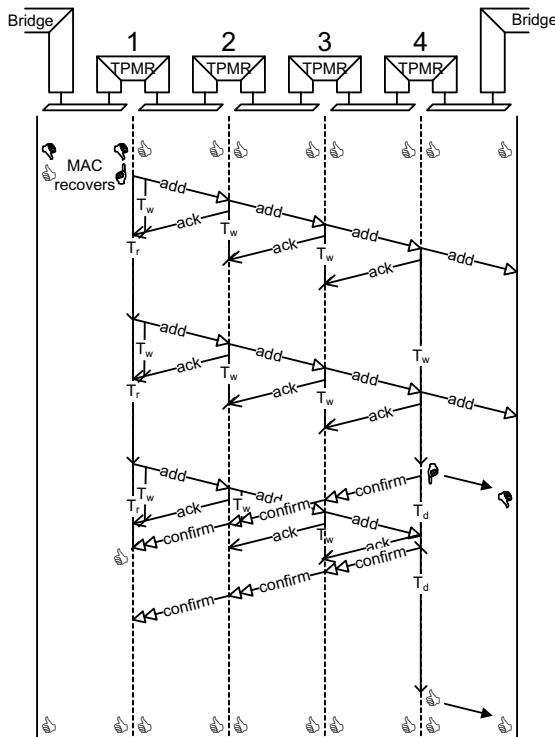


Figure K-4—Recovery at the end of a chain

If the LAN between the end system and the first TPMR in the chain (TPMR 1) had failed instead of recovering, the rest of Figure K-4 would have looked the same, with the exception of the final state of that first LAN. The net effect of the status notification protocol is to transfer the change in MAC_Operational to the other end of the chain so that both ends see the transition. For best effects the transfer should complete in a short enough time so that the end system protocols at the initiating end of the link are still executing their initial state.

Figure K-5 shows what happens if both ends of the link come up at the same time, the net effect is simply to delay the simultaneous start.

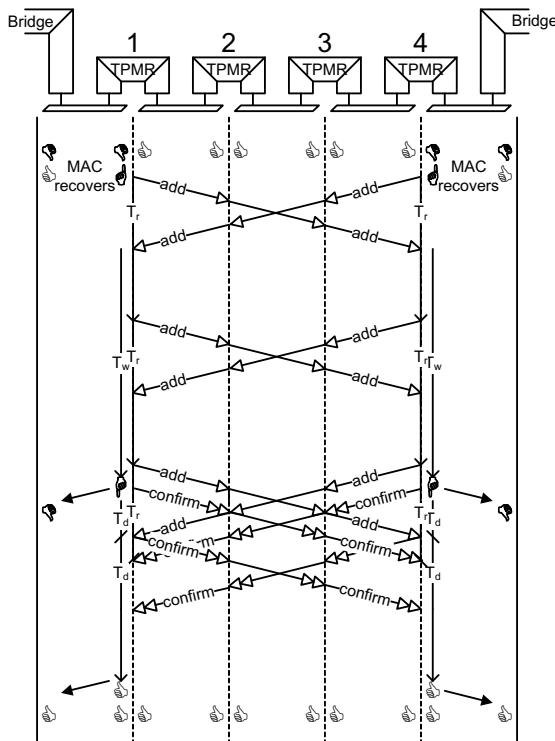


Figure K-5—Near simultaneous recoveries

Figure K-6 shows simultaneous transitions of one LAN in the chain to OperUp and the other to OperDown, and illustrates a number of points about how the protocol or protocols do or should operate.

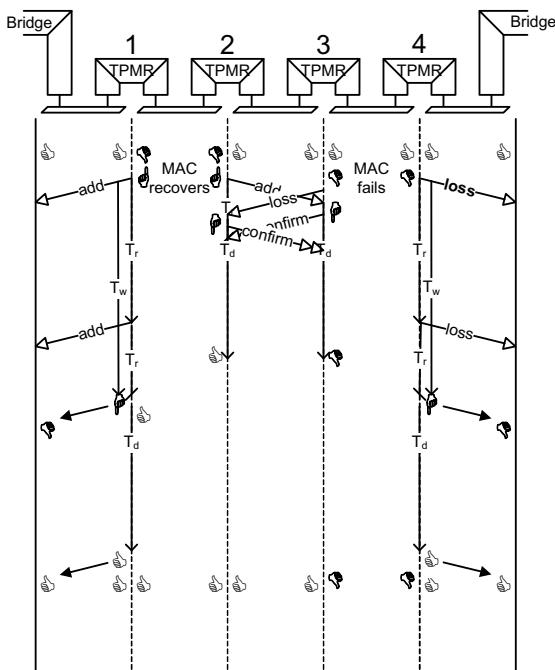


Figure K-6—Near simultaneous failure and recovery

First, signaling of addition or loss can be somewhat arbitrary and depend on the relative timing of transitions on separate individual LANs. The same final connectivity can be represented by a loss followed by an addition, or an addition followed by a loss. If the implied resulting state is to mean anything then the information “connected as far as,” or “connected to and beyond” also needs to be communicated. However, while this is a candidate for inclusion in any new protocol, support by existing protocols is unlikely. However this clause persists with distinguishing Add and Loss messages, and their confirmations, because that ensures that closely spaced transitions, which might otherwise disguise unusual loss of higher layer protocol messages, are not missed by the systems connected to the TPMR chain. Figure K-7 shows a loss followed by recovery of the same LAN.

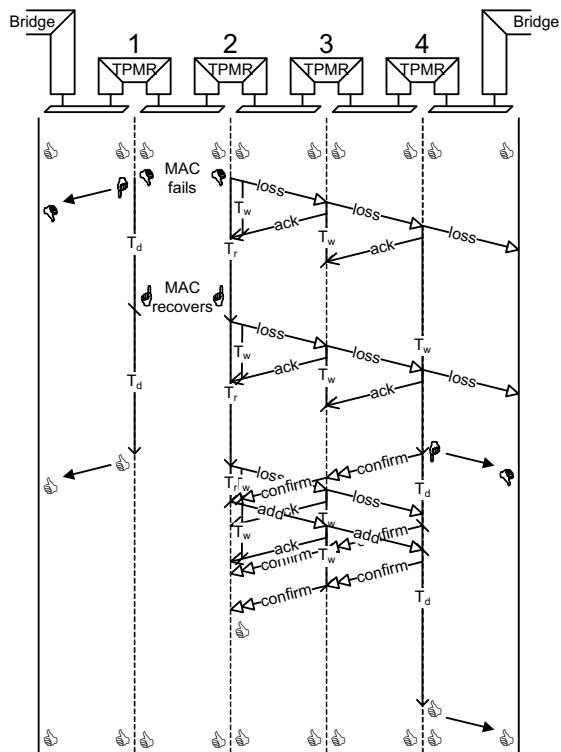


Figure K-7—Loss with quick recovery

Second, no message is sent across the LAN between TPMRs 1 and 2 when the MAC is OperDown from the protocol clients point of view. That allows support of the protocol to be architected such that a message that signals “add(ition)” or “loss” is forwarded through the normal bridged path, so its speed of propagation does not depend on scheduling or notifying a status propagation process in each TPMR. This also means that we have a free choice as to what additional functionality can reside between the entities responsible for status propagation and the “real” MAC.

One last common case of simultaneous transitions is what happens when the two ports of a TPMR power up or are enabled at the same time. Each tries to send a link notification through the other, and each link notification is discarded as the Port is not yet available (`mssOperUp` FALSE). The result is to “blip” each LAN. The ends of the TPMR chain see an Add Loss Add sequence, if they are protocol capable, and a blip in MAC Operational otherwise. If T_w is less than T_d , this will be a single blip.

Annex L

(informative)

Operation of the credit-based shaper algorithm

This annex contains a more detailed analysis of the way the credit-based shaper algorithm (8.6.8.2) operates, and how its operation affects the performance of the network, from the point of view of traffic that uses a credit-based shaped queue, and also from the point of view of other queues associated with the Port.

L.1 Overview of credit-based shaper operation

The credit-based shaper algorithm has a single externally determined parameter, *idleSlope* (see 8.6.8.2), that determines the maximum fraction of the *portTransmitRate* that is available to the queue associated with a traffic class (*bandwidthFraction*), as follows in Equation (L.1):

$$\text{bandwidthFraction} = \text{idleSlope}/\text{portTransmitRate} \quad (\text{L.1})$$

where *portTransmitRate* is the maximum transmit data rate that the port supporting the outbound queue is able to deliver. The detailed derivation of Equation (L.1) can be seen in Equation (L.5) through Equation (L.8).

If a traffic class supported by the credit-based shaper algorithm uses less than the bandwidth allocated to it, then the unused bandwidth can be used by other traffic classes, in accordance with the relative priorities of the traffic classes and the transmission selection algorithms that are associated with them.

NOTE 1—It is a natural consequence of the way transmission selection operates that if a given traffic class does not have a frame available for transmission, then a lower priority traffic class that does have a frame available for transmission is able to transmit. Hence, if a given traffic class that uses the credit-based shaper algorithm has been allocated X% of the available bandwidth on a Port, but only uses (X-Y)% of that allocation, then the unused portion of its allocation (Y%) is available for other traffic classes to use if they are in a position to do so. Whether this unused bandwidth can be used by a traffic class depends upon the transmission selection algorithm that is associated with it. For example, a traffic class that has been allocated a specific bandwidth, such as one that uses the credit-based shaper algorithm, would be unable to use any bandwidth allocation not used by a higher priority traffic class, as that would result in it exceeding its bandwidth allocation; however, a traffic class that uses the strict priority algorithm, which is not associated with a bandwidth allocation, would be able to use bandwidth allocated to, but not used by, a higher priority traffic class.

The *idleSlope* parameter, in conjunction with the size of the frames that are being transmitted using the queue, and the maximum time delay that a queue can experience before it is able to transmit a queued frame, places an upper bound on the burst size that can be transmitted from queues that use the algorithm.

In order to describe the operation of the algorithm, it is useful to define the following values, in addition to the formal parameters of the algorithm described in 8.6.8.2:

- a) ***maxFrameSize***. The maximum sized frame that can be transmitted through the Port for the traffic class concerned. This value can be determined by admission control algorithms, or can be simply a function of the transmission behavior of the traffic source. Either way, the value can be smaller than would be allowed by the normal operation of the underlying MAC service.
- b) ***hiCredit***. The maximum value that can be accumulated in the *credit* parameter. This parameter is a function of the operation of the algorithm, and cannot be controlled by management.
- c) ***loCredit***. The minimum value that can be accumulated in the *credit* parameter. The value of *loCredit* is a function of the values of *maxFrameSize*, *portTransmitRate*, and *sendSlope*, as follows in Equation (L.2).

$$loCredit = maxFrameSize \times (sendSlope/portTransmitRate) \quad (L.2)$$

- d) ***maxInterferenceSize***. The maximum size, in bits, of any burst of traffic that can delay the transmission of a frame that is available for transmission for this traffic class. For the numerically highest traffic class, *maxInterferenceSize* is exactly equal to the maximum sized frame that can be transmitted through the Port (the maximum frame size supported by the underlying MAC). For all other traffic classes, the derivation of *maxInterferenceSize* becomes more complex, owing to the possibility that any higher traffic classes that support the credit-based shaper algorithm can delay the transmission of a frame. *maxInterferenceSize* must also account for any media access delay, such as the recovery time in Energy Efficient Ethernet. A detailed analysis can be found in L.3.1.1.

NOTE 2—The definition of *maxInterferenceSize* assumes that any traffic classes that support the shaper algorithm are numerically higher than any traffic classes that support the default strict priority algorithm (8.6.8.1). If this is not true, then the value of *maxInterferenceSize* is infinite, and the operation of the credit-based shaper algorithm cannot provide the bandwidth that has been reserved for it. A detailed analysis of how *maxInterferenceSize* can be determined for a given traffic class is contained in L.3.

The value of *hiCredit* is determined by the worst case interfering traffic, as follows in Equation (L.3).

$$hiCredit = maxInterferenceSize \times (idleSlope/portTransmitRate) \quad (L.3)$$

The value of *hiCredit* can therefore be considered as a “high water mark”; the operation of the algorithm is such that the value of the *credit* parameter will never exceed *hiCredit*.

The choice of value for *idleSlope*, and the calculated value of *hiCredit*, can be used to determine the maximum burst size that can be output from the queue, as follows in Equation (L.4).

$$maxBurstSize = (portTransmitRate \times ((hiCredit - loCredit)/(-sendSlope))) \quad (L.4)$$

NOTE 3—There is an interrelationship between the maximum burst size and the buffer size available to the traffic class. There is no point in accumulating more credit than the traffic class can handle in queued frames, as the limiting factor is the discarding of frames when the queue overflows. This means that the amount of bandwidth that can be reserved on a given Port for a given traffic class depends upon the amount of buffering available to that traffic class. This in turn, combined with the port’s transmission rate, will determine the maximum value of idle slope that can be supported for that traffic class on that Port.

The operation of the credit-based shaper algorithm is illustrated in the examples shown in Figure L-1 through Figure L-3. In Figure L-1, a frame is queued at a time when the *credit* value is zero, and there is no conflicting traffic (there is no higher priority traffic awaiting transmission, and there is no frame being transmitted on the Port). The frame is immediately selected for transmission, and *credit* decreases at the rate of *sendSlope* as the transmission proceeds. Once the frame transmission is complete, *credit* increases back to zero at the rate of *idleSlope*, at which point, a further frame can be selected for transmission.

If a continuous stream of frames is made available to the shaper algorithm, i.e., there is always one frame queued awaiting transmission when the *credit* value reaches zero, then the fraction of the *portTransmitRate* that is available to the queue (*bandwidthFraction*) is equal to the fraction of time that frames are being transmitted from the queue. Assuming that *credit* decreases to value *loCredit* at the end of each frame transmission, then the following shows the detailed derivation of Equation (L.1):

$$bandwidthFraction = (-loCredit/sendSlope)/[(-loCredit/sendSlope) + (loCredit/idleSlope)] \quad (L.5)$$

$$= (-1/sendSlope)/[(1/idleSlope) - (1/sendSlope)] \quad (L.6)$$

$$= (-idleSlope)/(sendSlope - idleSlope) \quad (L.7)$$

$$= idleSlope/portTransmitRate \quad (L.8)$$

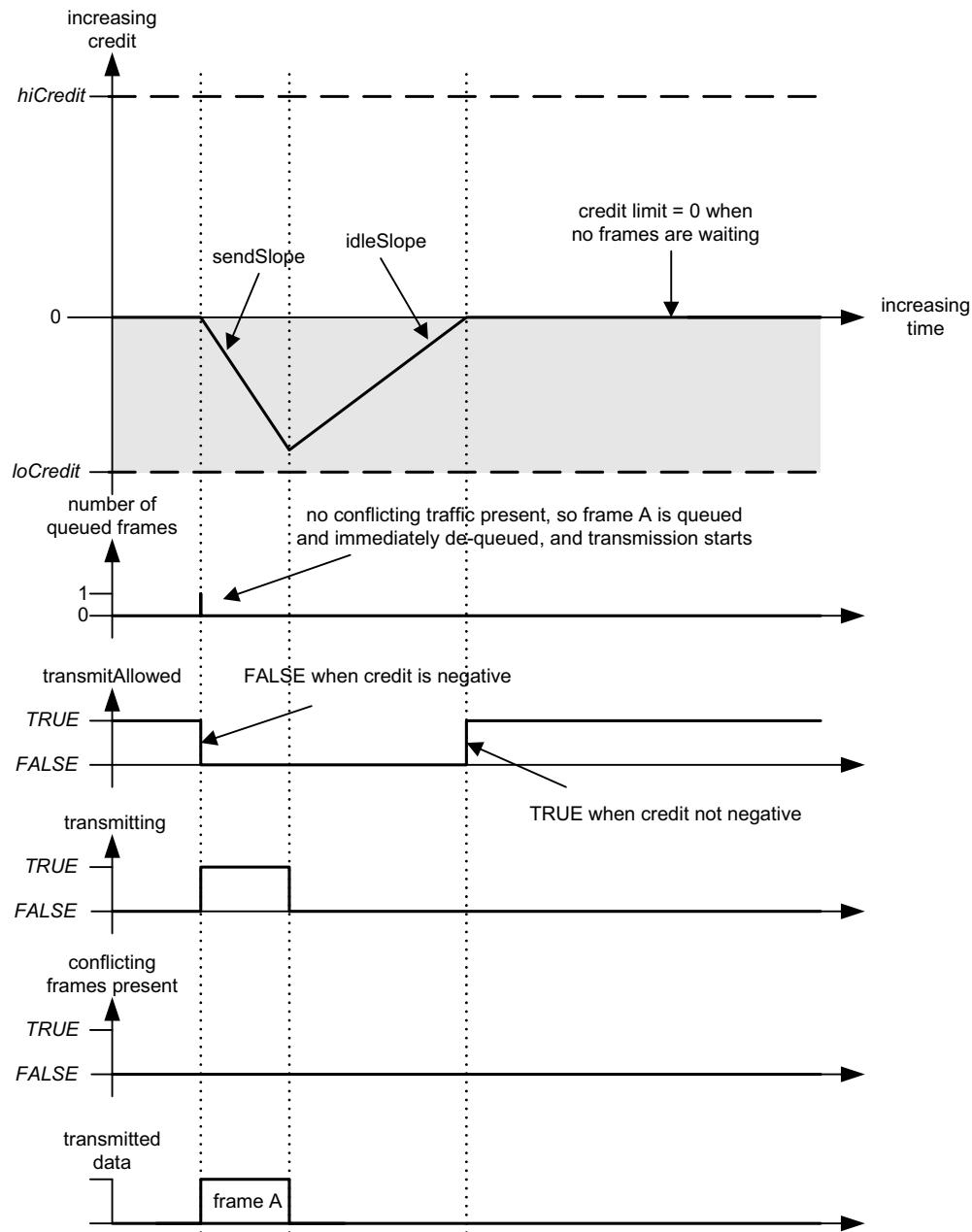


Figure L-1—Credit-based shaper operation—no conflicting traffic

This fraction of the bandwidth is available to the stream even in the presence of conflicting traffic, as illustrated in the following examples. In Figure L-2, a frame is queued at a time when the Port is transmitting conflicting traffic. The value of *credit* increases at the rate of *idleSlope* while the queued frame waits for the Port to become available. Transmission of the conflicting traffic completes before the value of *credit* is limited by *hiCredit*, and transmission of the queued frame starts. The value of *credit* starts to decrease at the rate of *sendSlope*; however, as the frame is not large enough to consume all of the available *credit*, and there are no further frames queued, *credit* is reduced to zero on completion of the transmission.

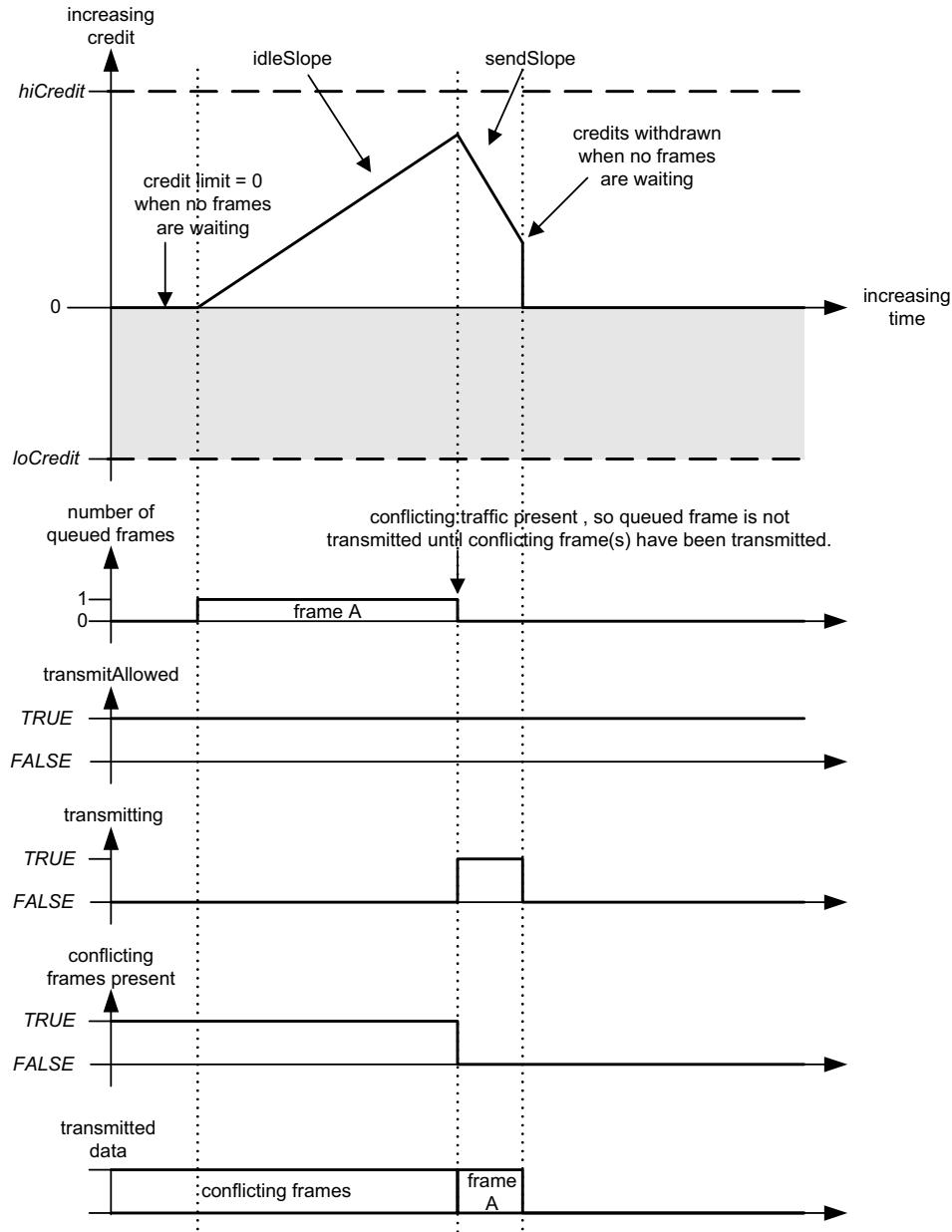


Figure L-2—Credit-based shaper operation—conflicting traffic

In Figure L-3, three frames are queued while the Port is transmitting conflicting traffic, and *credit* accumulates at the rate of *idleSlope*. Once the conflicting traffic has been transmitted, the first and second frames are transmitted back-to-back, because transmitting the first frame leaves *credit* ≥ 0 . However, as transmitting the second frame causes *credit* to become negative, transmission of the third frame is delayed until *credit* returns to zero.

L.2 “Class measurement intervals” in Bridges

The “class measurement interval” defines acceptable Talker behavior with respect to per-stream transmission (see 34.6.1); however, it does not directly impact the operation of the credit-based shaper algorithm in Bridges.

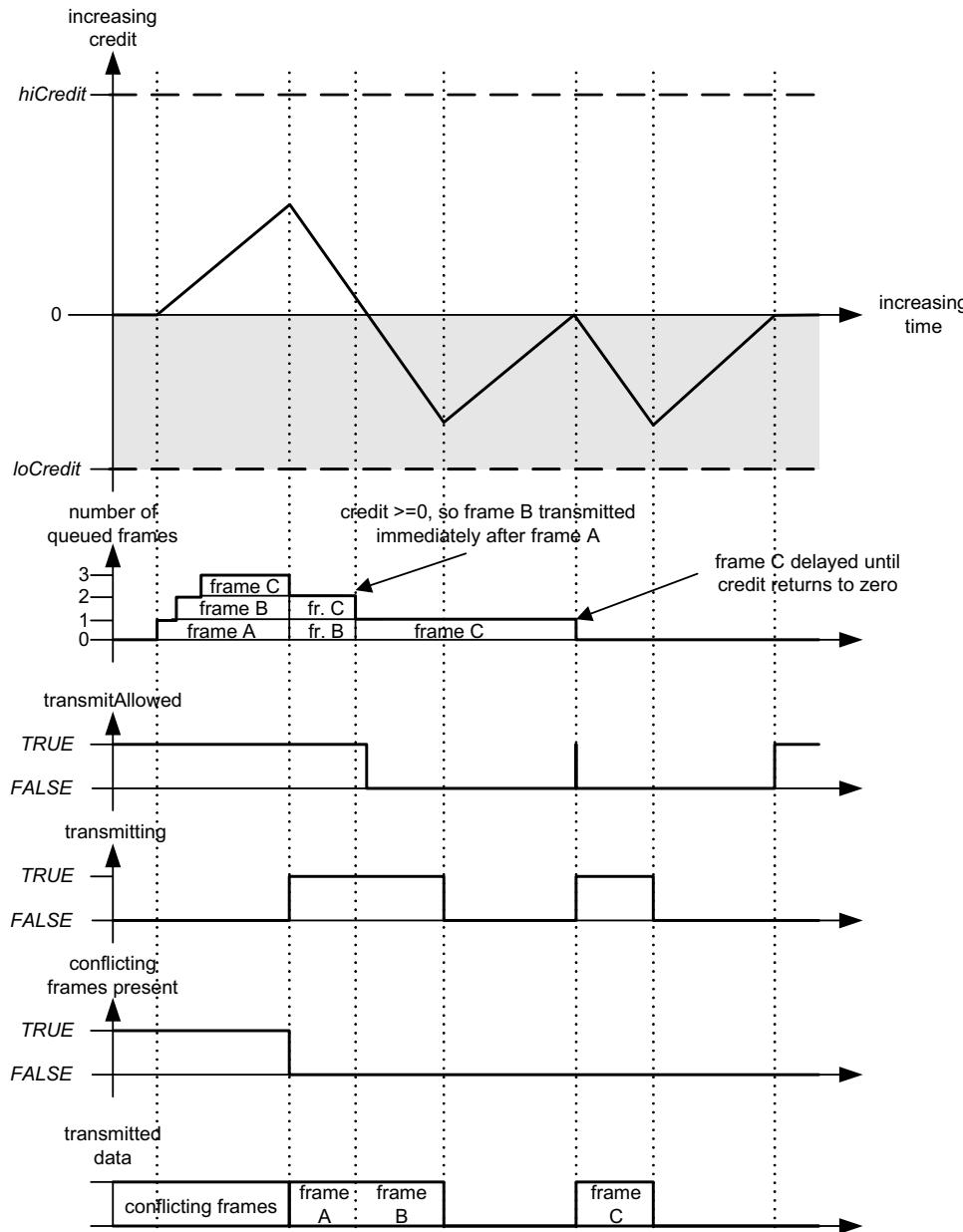


Figure L-3—Credit-based shaper operation—burst traffic

The maximum fraction of the available bandwidth that a queue operating the credit-based shaper algorithm consumes is determined by *idleSlope*, as discussed in L.1; if *idleSlope* is 75% of the link transmission rate, then the algorithm gets a maximum of 75% of the link, averaged over some time period (the class measurement interval) at the Talker. For a Bridge, where transmission becomes bursty due to interfering transmission from other traffic classes, the minimum measurement interval that is meaningful (i.e., the minimum period over which it is still possible to observe that the 75% maximum bandwidth is not exceeded) is the time taken for the traffic class concerned to transmit a maximum sized burst, multiplied by 100/75. In the case of the 125 μ s class measurement interval for SR class A traffic, on 100 Mbit/s Ethernet, and with 75% reservation, the approximate calculation is (ignoring interframe gaps):

- a) The maximum interference size (*maxInterferenceSize*) is one maximum sized Ethernet frame, which is 2000 octets, or 16000 bits, so *hiCredit* would be 75% of that, or 12000 bits.

NOTE 1—Media access delays, such as those imposed by Energy Efficient Ethernet, could be more significant than the interference created by a maximum sized frame.

NOTE 2—The figure of 2000 octets is a maximum for Ethernet and therefore a worst case for an interfering frame; in a specific implementation, the maximum frame size that is used could be smaller. In implementations where the maximum frame size is known to be less than 2000, the smaller value could be used.

- b) The maximum frame size for this traffic class is 75% of the number of bits that can be transmitted in 125 µs, so 75% of 12 500 bits in the case of a 100 Mbit/s LAN, or 9375 bits (including up to 500 or so possible “overhead” octets—used for Tag headers, physical layer overhead, etc.). As this is not an integral number of octets, this figure is rounded down to 9368. *loCredit* is 25% of that, or –2342.
- c) The maximum burst size, for frames no greater than 9368 bits long, would occur if *credit* reached 12000, then a series of frames were transmitted that exactly took credit to 0, followed by a final 9368 bit frame that would take credit down to –2342. This maximum burst is therefore calculated as: $(12\ 000 + 2342) \cdot (\text{portTransmitRate}/(-\text{sendSlope}))$ or 57 368 bits.
- d) The minimum measurement interval over which one can expect to be able to measure the bandwidth utilisation for SR class A, and observe that it does not exceed its 75% allocation, is therefore 57368 bit times multiplied by 100/75, which is 76 491 bit times, rounded up to the nearest integer, or 764.91 µs.

NOTE 3—The maximum sized burst will be preceded by a *maxInterferenceSize* event, and at the end of the burst, *credit* will have been reduced to –2342; it will therefore take $2342/\text{idleSlope}$ seconds, or nearly 3123 bit times, for *credit* to return to zero, and for SR Class A to be allowed to transmit again. In addition, the actual utilization for SR Class A is $57368/(57368+16000+(2342/0.75)) = 75\%$.

For the SR class B traffic, all that really changes is the potential size of any interference, which increases from the SR class A *maxInterferenceSize* to the maximum burst size for SR class A plus the class A *maxInterferenceSize*.

So, the smallest possible measurement interval over which the ratio *idleSlope/portTransmitRate* will be seen to hold good is dependent on the maximum interference size for the SR class, the maximum frame size for the SR class, and the value of *idleSlope*.

L.3 Determining worst-case latency contribution and buffering requirements

From the point of view of the operation of AV Bridges, it is important to be able to assess the contribution that each bridge makes to the worst case latency that stream data frames can be subject to during their transmission along the path from Talker to Listener. Closely related to the latency contribution that a Bridge makes is the buffering requirement that is required in the Bridge in order for the delay not to result in frame discard; if a Bridge can potentially delay a frame of a given SR class for a given period of time, then it must be capable of buffering the worst-case number of frames of that same SR class that could be received for onward transmission on each Port. The following discussion looks at the factors that contribute to the delay that could be experienced by a frame as it passes through a Bridge, and how the delay can be accurately determined.

NOTE 1—This discussion is also useful to guide designers of Talker stations in determining the delay added by the Talker’s network interface.

The worst case latency for a single hop from Bridge to Bridge, measured from arrival of the last bit at Port *n* of Bridge A to the arrival of the last bit at Port *m* of Bridge B, can be broken out into the following components:

- a) Input queuing delay. (There are no input queues in the IEEE 802.1 architecture, but if present, the implementation must account for them.)
- b) Interference delay (L.3.1).
- c) Frame transmission delay. (The time taken to transmit one maximum frame at *portTransmitRate*.)
- d) LAN propagation delay. (A variable delay that depends on the length of the LAN connection to the next Bridge; this is measurable using the mechanisms defined in IEEE Std 802.1AS [B8].)
- e) Store-and-forward delay. This includes all other elements of forwarding delay that are a consequence of the internal processing of the Bridge, assuming that the input and output queues are empty, such as:
 - 1) The time needed to pass a frame from the input port to the output port, assuming empty queues.
 - 2) The time delay between a frame being available for transmission on a Port and the Port being ready to transmit the frame.

NOTE 2—For example, in the case where the MAC/PHY has entered a power saving mode, there may be a delay incurred in switching the Port back to normal operation.

- 3) The difference, if any, in the delay incurred by a frame that bypasses an empty queue, vs. that incurred by a frame that must be enqueued.
- 4) The time added (subtracted) by the lengthening (shortening) of the frame due to addition (removal) of frame headers such as Q-tags or MACSec-tags.
- 5) The time needed to encrypt a MACSec frame.

In the remainder of this clause, the following abbreviations are used:

M_0	Maximum sized frame for non-SR classes
M_x	Maximum sized frame for SR class x
R_0	<i>portTransmitRate</i>
R_x	idleSlope for SR class x
T_{xy}	Time interval between events x and y
W_x	-sendSlope _x
$variable_{<x}$	The sum of the values of the named <i>variable</i> for all SR classes with a higher priority (and therefore earlier letter in the collating sequence) than x

L.3.1 Interference delay

The interference delay for frame X can be broken out into the following components:

- a) **Queuing delay:** The delay caused by the frame that was selected for transmission an arbitrarily small time before frame X became eligible for transmission selection, plus the delay caused by queued-up frames from all stream frames with higher priority than frame X's class (i.e., the "maxBurstSize" for SR Classes with higher priority than X—see L.1). This is what is referred to as *maxInterferenceSize* in L.1. Queuing delay is analyzed in detail in L.3.1.1.
- b) **Fan-in delay:** The delay caused by other frames in the same class as frame X that arrive at more-or-less the same time from different input Ports. Fan-in delay is analyzed in detail in L.3.1.2.
- c) **Permanent delay:** The delay caused by frames that reside in a buffer for a long time, relative to the output queuing delay, because of the history of activity in the network. Permanent delay is analyzed in detail in L.3.1.3.

L.3.1.1 Queuing delay

The queuing delay for frame X can be broken out into the following components:

- a) The delay, *maxInterferenceTime*, before a frame X queued for an SR class, and that is eligible for transmission, can actually be transmitted. This delay is experienced when a frame, of any priority, is selected for transmission an arbitrarily small time before frame X became eligible for transmission; it is also experienced in LAN media, such as Ethernet, where there can be a low energy consumption mode that is entered under idle conditions, and where there is a significant delay, *mediumAccessDelay*, before the LAN returns to normal operation and is able to transmit. The value of *maxInterferenceTime* is therefore equal to whichever is the greater of the following:
 - 1) The time taken to transmit the maximum frame size supported by the medium, i.e., $\text{maxFrameSize}_{(m)} / \text{portTransmitRate}$.
 - 2) The *mediumAccessDelay* for the medium.
- b) The delay caused by any queued-up frames associated with the next higher priority SR class than frame X's SR class (i.e., *maxBurstSize* for the next higher priority SR class), if frame X does not belong to the highest SR class.

For SR class A, only the first of these components applies, as there is no higher SR class.

Suppose that the queue for SR class A is full, and it has accumulated the maximum amount of *credit* (*hiCredit*). Because SR class A frames have priority over all other traffic (even BPDUs), *hiCredit* for SR class A is merely the credit accumulated during the *maxInterferenceTime* required to transmit a lower-priority frame. Until that accumulated credit is exhausted, SR class B (C, D,...) frames cannot be transmitted.

If SR class A were permitted to use 100% of the LAN bandwidth, then the SR class A queue would never catch up, because it would use credit as fast as it was gained, and there would never be a chance for any other traffic class to transmit a frame. If SR class A were permitted to use 99% of the LAN bandwidth, then SR class A can transmit back-to-back frames and the accumulated *credit* would be drained at 1% of the LAN bandwidth until it goes negative; at that point, SR class A cannot transmit further frames until its *credit* returns to a non-negative value, which ensures that at least 1% of the bandwidth is available to other traffic classes.

NOTE—In general, if SR Class A is permitted to use X% of the bandwidth, then only (100-X)% remains available for use by lower priority traffic classes. In practice, as 75% is the default value for the highest percentage of *portTransmitRate* that can be used by all SR classes, there is always at least 25% of *portTransmitRate* available for use by the non-SR traffic classes.

Figure L-4 illustrates this burst transmission behavior, and the effect of *maxInterferenceTime* on the latency of SR class frames. At point *a* on the time line, the queues for SR classes A and B are empty, and a maximum sized frame (M_0 bits long) belonging to a non-SR traffic class starts transmission. An instant later, SR class A and B frames arrive that are all of the maximum size that the classes are currently experiencing (M_A and M_B bits long, respectively); however, as there is a frame already being transmitted, they are queued. At point *b*, transmission of the non-SR class frame completes, and the first of a burst of four SR class A frames are transmitted. By point *d*, SR class A's credit has been taken negative by the transmission of the last of its burst of frames, and therefore no further SR Class A frames can be transmitted until its credit is no longer negative; this allows SR Class B to transmit a frame. What happens at the end of the class B frame will depend on what frames are queued waiting for transmission, and whether the credit available to either of the SR classes is non-negative.

The queuing delay experienced by the first SR Class A frame is therefore the time interval between *a* and *b*, T_{ab} . If R_0 is the transmission rate (*portTransmitRate*), then:

$$T_{ab} = M_0 / R_0 = \text{maxInterferenceTime} \quad (\text{L.9})$$

Figure L-5 shows how credit changes for SR Class A during this sequence.

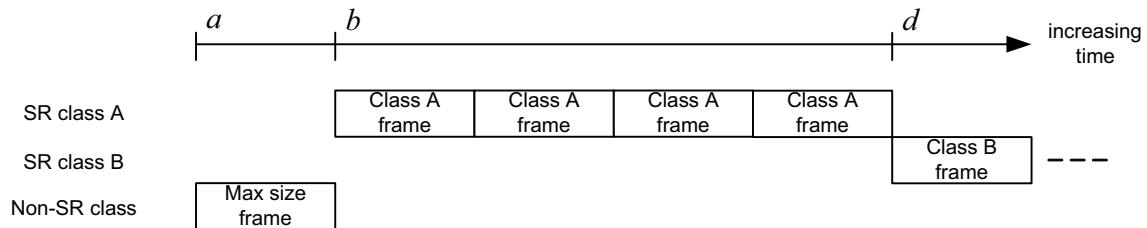


Figure L-4—Interference and latency

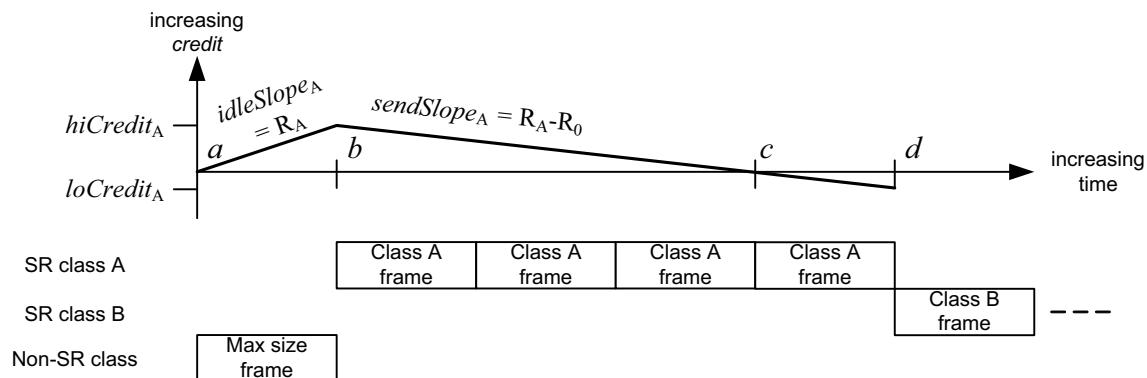


Figure L-5—Burst behavior and credit

R_A is the reserved data rate for SR Class A (which is the *idleSlope* for class A), R_B is the reserved data rate for SR Class B, and so on. The value of *credit*_A accumulates up to *hiCredit*_A during the transmission of the maximum sized non-SR frame, so:

$$hiCredit_A = R_A \times M_0 / R_0 = idleSlope_A \times maxInterferenceTime \quad (L.10)$$

credit is drained during the transmission of the SR Class A frames, at the rate *sendSlope*_A:

$$sendSlope_A = (R_A - R_0) \quad (L.11)$$

credit reaches zero at point *c* in Figure L-5, which, for the purposes of this discussion, is assumed to happen exactly at the point where transmission of the third class A frame finishes. Since a frame can be transmitted when *credit* = 0, Class A's credits continue to drain to the value:

$$loCredit_A = (R_A - R_0) \times M_A / R_0 \quad (L.12)$$

during the transmission of the fourth class A frame (M_A bits long, transmitted in time M_A / R_0). At point *d*, transmission of the fourth frame completes; as *credit*_A is now negative, no more class A frames can be transmitted, and transmission of a class B frame starts.

So, for class A, the maximum burst size is equal to the transmission rate, R_0 , multiplied by the time interval between points *b* and *d* in Figure L-5:

$$\text{max Class A burst size} = R_0 \times (- (hiCredit_A - loCredit_A) / sendSlope_A) \quad (L.13)$$

Substituting *hiCredit*_A, *sendSlope*_A, and *loCredit*_A from Equation (L.10), Equation (L.11), and Equation (L.12), respectively:

$$\max \text{ Class A burst size} = R_0 \times (- (R_A \times M_0 / R_0 - (R_A - R_0) \times M_A / R_0) / (R_A - R_0)) \quad (\text{L.14})$$

$$= R_0 \times (R_A \times M_0 / R_0) / (R_0 - R_A) + M_A / R_0 \quad (\text{L.15})$$

$$= (R_A \times M_0) / (R_0 - R_A) + M_A \quad (\text{L.16})$$

The queuing delay experienced by SR class B is the time interval between a and d in Figure L-5, T_{ad} :

$$T_{ad} = T_{ab} + T_{bc} + T_{cd} \quad (\text{L.17})$$

$$= M_0 / R_0 + hiCredit_A / sendSlope_A + M_A / R_0 \quad (\text{L.18})$$

$$= M_0 / R_0 + (R_A \times M_0 / R_0) / (R_0 - R_A) + M_A / R_0 \quad (\text{L.19})$$

$$= M_0 / (R_0 - R_A) + M_A / R_0 \quad (\text{L.20})$$

This can be generalized to calculate the queuing delay experienced by any SR class, X, by using the sum of the credits available to all higher priority SR classes. In the following, the subscript “ $<X$ ” is used to indicate the sum of the values for all SR classes with higher priority (and therefore, earlier letters in the collating sequence) than X.

The credit acquisition rate $idleSlope_{<X}$ for SR classes A through X-1 is the sum of the data rates for all higher priority classes, so:

$$idleSlope_{<X} = \sum_{K < X} R_K \quad (\text{L.21})$$

This is just the sum of the $idleSlope_K$ values for the higher classes. However, the combined classes accumulate credits only until the single interfering non-SR interfering frame stops transmitting, and then the credits start decreasing linearly. So, the upper limit for the combined credits is not the sum of the individual Class's credits; it is the number of bits divided by the slope, or:

$$hiCredit_{<X} = (\sum_{K < X} R_K) \times M_0 / R_0 \quad (\text{L.22})$$

Similarly, the combined $sendSlope_{<X}$ is:

$$sendSlope_{<X} = -(R_0 - \sum_{K < X} R_K) \quad (\text{L.23})$$

These combined rates hold true until some class's buffer empties and its credits are forced to 0. In the worst-case scenarios we are examining, this does not happen.

Defining

$$W_{<X} = -sendSlope_{<X} = R_0 - \sum_{K < X} R_K \quad (\text{L.24})$$

we have:

$$sendSlope_{<X} = -W_{<X} \quad (\text{L.25})$$

and

$$hiCredit_{<X} = (R_0 - W_{<X}) \times M_0 / R_0 \quad (\text{L.26})$$

For all combined classes, T_{ab} is the same as for SR class A, the time for the non-SR frame to transmit:

$$T_{ab} = M_0 / R_0 \quad (L.27)$$

The combined classes A through X-1 drain from $hiCredit_{$ to 0 in time $hiCredit_{} / W_{, so:$

$$T_{bc} = ((R_0 - W_{$$

The worst case value of $loCredit_{ for the combined $<X$ classes is not the sum $\sum_{K<X} loCredit_K$ as this would overestimate the worst case. Only one of the $<X$ classes can be transmitting a frame at a given time, so the end of the burst for the $<X$ classes occurs when one of the $<X$ classes, Q say, starts transmitting a frame at a point where all the other $<X$ classes have negative credit, and their credit is therefore rising. If, by the time Q's frame transmission finishes, the other $<X$ classes still have negative credit, then Q's frame is the last frame of the $<X$ burst; but at that point, all the other $<X$ classes must, by definition, have more credit than their respective $loCredit$ values (i.e., for these other classes, $credit > loCredit$) because even if they were at their respective $loCredit$ values when Q started transmitting, they have been gaining credit during the transmission of Q's frame.$

It is also not possible to decide that $loCredit_{ is simply the time needed to transmit one copy of each Class's maximum-length frame; for some classes it could be possible transmit more than a single last frame after the total credits = 0, even if all Classes' credits reach 0 simultaneously. For example, classes A and B reach $credit = 0$ at the same time; A transmits a frame, then B transmits a frame. By the end of the B frame, A's $credit$ has returned to zero, and so A can transmit a further "last" frame.$

Some class must transmit the last frame, and for it to be the worst case, that last frame is a maximum length frame. If it were not, then either:

- c) Extending it to the maximum length still leaves the other classes at negative $credit$; or
- d) Extending it to the maximum length leaves one or more other class with 0 or positive $credit$, in which case they will transmit more frames.

So, for class Q to be the class that transmits the last frame of the $<X$ burst, the following condition must be true for all other $<X$ classes, P:

$$M_q \times R_p > (R_0 - R_p) \times M_p \quad (L.29)$$

Equation (L.29) follows from the fact that the increase in credits of class P during the transmission of M_q , which is equal to $(M_q/R_0) \times R_p$ is greater than the absolute value of the $loCredit$ of P, which is equal to $(R_0 - R_p) \times M_p$. So:

$$(M_q/R_0) \times R_p > (R_0 - R_p) \times M_p / R_0 \quad (L.30)$$

This statement is true because the other classes P must have low enough $credit$ that their $credit$ cannot climb above 0 during the transmission of M_q .

The very lowest that $credit_{ might reach for three SR classes, Q, P, and S is shown in Equation (L.31).$

$$loCredit_{$$

However, even this is pessimistic, as it assumes that more than one class can be at their $loCredit$ at the same time, which is not the case. What this shows, however, is that if R_K is small for all $K < X$ (i.e., $R_K \ll R_0$), then the $R_K \times M_q/R_0$ terms all approach zero, and therefore, $loCredit_{ approaches the sum $\sum_{K < X} loCredit_K$. So:$

$$loCredit_{$$

$$= \sum_{K < X} (R_K - R_0) \times M_K / R_0 \quad (L.33)$$

But in this worst case, $R_K \ll R_0$, so:

$$loCredit_{<X} = \sum_{K < X} (-M_K) \quad (L.34)$$

Putting together these various components, the queuing delay for SR Class X, $qDelay_X$, is as follows:

$$qDelay_X = M_0 / R_0 + (hiCredit_{<X} - loCredit_{<X}) / |sendSlope_{<X}| \quad (L.35)$$

$$= M_0 / R_0 + ((R_0 - W_{<X}) \times M_0 / R_0 + \sum_{K < X} M_K) / W_{<X} \quad (L.36)$$

$$qDelay_X = (M_0 + \sum_{K < X} M_K) / W_{<X} \quad (L.37)$$

For SR class A, this equation reduces to Equation (L.38):

$$qDelay_A = M_0 / W_{<A} \quad (L.38)$$

As there is not a “<A” class, A being the highest priority class, $W_{<A}$ is simply R_0 , so this reduces to Equation (L.39):

$$qDelay_A = M_0 / R_0 \quad (L.39)$$

This agrees with the earlier discussion in L.1.

For SR class B, the equation reduces to Equation (L.40):

$$qDelay_B = (M_0 + M_A) / W_A \quad (L.40)$$

As W_A is $-sendSlope_A$, which is just $R_0 - R_A$, this can be rewritten as shown in Equation (L.41):

$$qDelay_B = (M_0 + M_A) / (R_0 - R_A) \quad (L.41)$$

The maximum burst size for a given class, X, is the number of bits that can be transmitted in “back-to-back” frames by class X. The maximum burst occurs when that class has experienced the maximum queuing delay for that class, $qDelay_X$, and class X is then able to transmit frames, uninterrupted by any higher priority classes, until it reaches $loCredit_X$. The size of this burst, $maxBurst_X$, is equal to the number of bits that can be transmitted at line rate, R_0 , in the time it takes for creditX to reduce from $hiCredit_X$ to $-loCredit_X$, as follows in Equation (L.42):

$$maxBurst_X = (hiCredit_X - loCredit_X) \times R_0 / W_X \quad (L.42)$$

From the earlier results, the worst case for $loCredit_X$ is just $(-M_X)$, so:

$$= hiCredit_X \times R_0 / W_X + M_X \times R_0 / W_X \quad (L.43)$$

and $hiCredit_X$ is the amount of credit that can be accumulated during $qDelay_X$ seconds, which is $qDelay_X$ multiplied by the idle slope, R_X , so:

$$maxBurst_X = (M_0 + \sum_{K < X} M_K) \times R_X \times R_0 / (W_{<X} \times W_X) + M_X \times R_0 / W_X \quad (L.44)$$

L.3.1.2 Fan-in delay

The delay caused by other frames in the same class as frame X that arrive at more-or-less the same time from different input Ports. Consider the scenario in Figure L-6, where a Bridge has a number of Ports on which it receives frames for a given SR Class and the frames will be queued on a single outbound Port. Imagine that the “upstream” Bridge Ports feeding every reception Port of the Bridge all encounter a maximum interference event for Class X at the same time. The output Port will be starved of data. Then, once the interference events complete, the “upstream” Bridges start to transmit frames and the input Ports all receive a frame at the same moment; all received frames are delivered to the output queue.

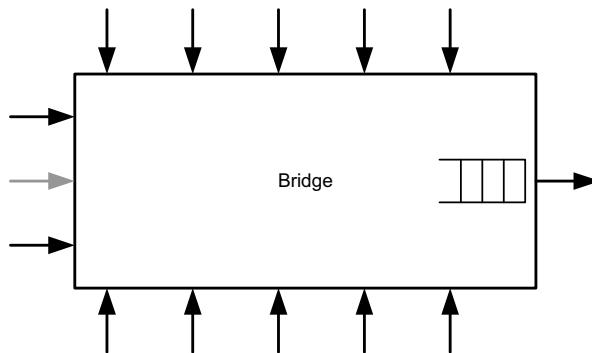


Figure L-6—Fan-in scenario

Assuming that all of the Ports of the Bridge support the same data rate, and the outbound queues of the upstream Bridges are all configured the same as the outbound queue of this Bridge, then the outbound Port would not be able to transmit frames from its queue at the desired, combined data rate of the incoming traffic, as this would require the outbound Port (in this example) to transmit at 20 times its reserved data rate, which would not be permitted by the Stream Reservation Protocol. It follows that whatever the reserved data rate is on the outbound Port, this must be the same as the sum of the reserved data rates being fed into the inbound Ports.

By allocating the majority of the reserved bandwidth to one of the inbound Ports (shown in grey in Figure L-6) and a very small amount of bandwidth to all of the other inbound Ports, the amount of fan-in data that will be received can be maximized; the grey Port will receive maxBurst_X bits of burst data, and all other Ports will receive a single frame of length M_X . Hence, the data that the outbound queue has to handle in this scenario is $\text{maxBurst}_X + 12 \times M_X$ bits of fan-in data.

NOTE 1—The reason that this choice maximizes the fan-in data burst size is that regardless of how little bandwidth is allocated to the black Ports, they still get to burst a complete frame.

The following is a procedure (not a formula) for determining the maximum amount of fan-in data for a given SR class and output port:

- a) Determine B_O , the maximum possible bandwidth for class X on the output port.
- b) For each possible input port i , determine the maximum possible bandwidth B_i for Class X—it is assumed that this information is obtained, by some means, from the transmitting Bridge that is the source of the frames received by i .
- c) For input port i , calculate $\text{maxBurst}_{X,i}$ using $\max(B_O, B_i)$ for the bandwidth. (In the maxBurst formula, use $W_X = R_0 - \max(B_O, B_i)$.)
- d) Add $\max(\text{maxBurst}_{X,i})$ to the total fan-in data.
- e) Set $B_O = B_O - B_i$ and repeat from step c) until $B_O = 0$.
- f) For each remaining port, add $M_{X,i}$ to the total fan-in data.

The fan-in delay is then equal to the time taken for the total fan-in burst data, computed as shown above, to be output at the line rate of the output port.

NOTE 2—The definition of a measurement interval for stream traffic specifications implies a minimum frame transmission rate (8000 frames per second for class A's 125 us measurement interval). At this frame transmission rate, the frames are smaller and, due to the overhead in small frames, fewer streams are possible (no more than 13 through a 100 Mb port). These constraints should be given due consideration when determining the value to be used for M_X and for determining the maximum number of ports contributing to fan-in delay.

Example calculations of fan-in delay can be found in the paper “Calculating the Delay Added by Qav Stream Queue” [B4].

L.3.1.3 Permanent delay

Permanent delay reflects the fact that, if a path through the network is being used continuously at its full capacity for a particular SR class, interfering traffic can cause “permanent” buffer occupancy at the point of congestion. This is illustrated in Figure L-7 through Figure L-10 and the accompanying text.

In Figure L-7 we have a Talker station, T, that has reserved the highest possible bandwidth, R_X , for an SR class X stream on the path through 3 Bridges to Listener L. For the purposes of the discussion, R_X is the maximum reservable data rate for class X on the outbound Ports of all three Bridges, and Talker T starts transmitting the stream, at the reserved data rate R_X . Assuming that the LANs are short and there is no interfering traffic (i.e., the only traffic being handled is this particular stream), the buffer occupancy in each Bridge will approach zero, as frames can be transmitted onwards as soon as they are received.

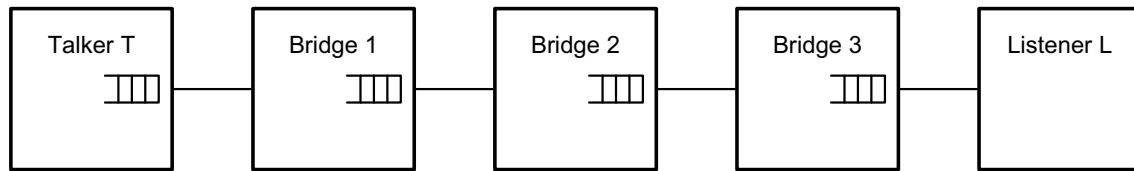


Figure L-7—Permanent delay scenario

In Figure L-8, a second transmission, Q, generates the maximum possible interference to class X (i.e., class X's next frame for transmission is delayed by $qDelay_X$). Talker T's class X queue fills, and its *credit* increases to $hiCredit_X$ (shown on the figure as $C=C_X$). Because T is no longer transmitting class X frames, the downstream Bridges are starved of frames to transmit, their buffers are empty, and their credit is zero ($C = 0$).

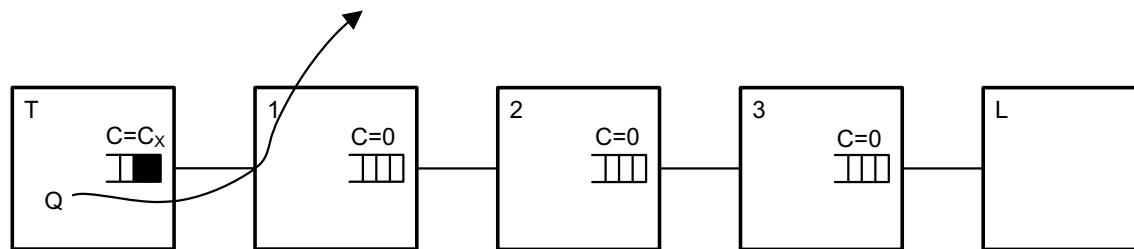


Figure L-8—Building up buffer occupancy—1

In Figure L-9, the interfering traffic goes away, and T is able to transmit a maximum sized class X burst of frames at line rate; once the burst has finished, T reverts to transmitting the stream at the reserved rate R_X . As Bridge 1 has no credit, it will have to buffer most of the burst of frames, and as T is sending frames as fast as Bridge 1 can get rid of them, its buffer occupancy will remain at that level unless something else

affects that steady state. Bridge 1 ends up with a credit value that hovers around the *hiCredit* level; the credit in the other Bridges will have *credit* of 0 or some value between 0 and *loCredit* as the Bridges receive and transmit frames.

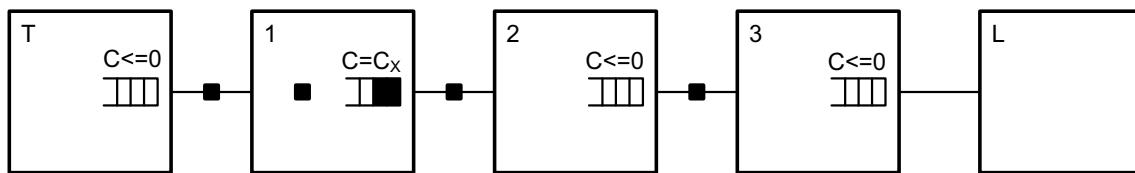


Figure L-9—Building up buffer occupancy—2

If the interfering traffic, Q, starts up again, Talker T's buffer occupancy will build up again; however, as Bridge 1 still has frames buffered, it can continue to transmit, and the downstream Bridges are not starved of frames, and Bridge 1's buffer occupancy and *credit* returns to around 0. When the interference stops, T transmits a burst at line rate, as before, which simply restores Bridge 1's queue to the state shown in Figure L-9.

If a new interfering frame source, S, starts up that affects class X in Bridge 1, then Bridges 2 and 3 could again be starved of traffic, and the buffer occupancy and credit in Bridge 1 could increase temporarily to $2C_X$ (see Figure L-10). When S stops transmitting, then Bridge 2 can create a line rate burst of traffic that will restore its buffer to the previous steady state (Figure L-9), but now Bridge 2 has a permanent buffer loading as shown in Figure L-10.

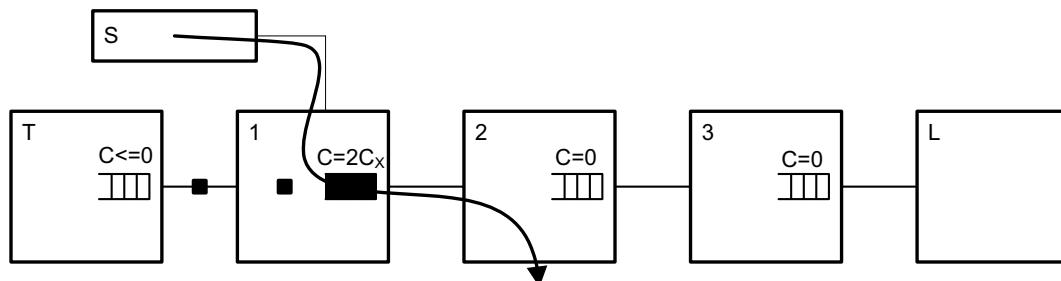


Figure L-10—Building up buffer occupancy—3

This can of course happen for other Bridges in the path, leading eventually to some level of permanent buffer occupancy in each Bridge (Table L-11). The problem does not get any worse from that point on, because no Bridge ever starves (i.e., no Bridge's credit is set to zero because it runs out of frames to send); starvation is a requirement in order for permanent partial occupancy to be created.

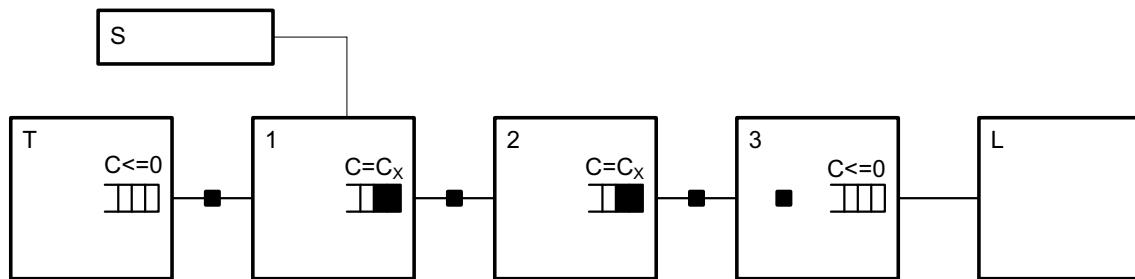


Figure L-11—Building up buffer occupancy—4

It may, therefore, be considered desirable for all Bridges to reserve a little bit more bandwidth than the actual total reserved by the Talkers, so that this “permanent” buffer build-up can drain away. However, this would not make the problem significantly better on the timescales of the output queuing delay, so it does not factor into those calculations, and, of course, this additional capacity would increase the queuing delay at each hop.

NOTE—Many applications have media clocks that are asynchronous to the free-running clock used to determine class measurement intervals. As the two clocks could be operating at the extremes of their permissible variation, it is necessary to reserve more bandwidth than is nominally needed in order to account for the potential difference in clock rate.

Since this kind of event could happen on multiple input ports at the same time, the “permanently buffered” data equals the worst-case fan-in data.

L.3.2 Maximum interference delay and maximum buffer requirement

As discussed in L.3.1, the worst-case interference delay for SR class X on a given Port is the sum of the following three parts:

- a) $qDelay_X$ [see Equation (L.35) in L.3.1.1]
- b) The fan-in delay (see L.3.1.2)
- c) The “permanent” buffer contents contribution (see L.3.1.3), which is equal to the worst-case fan-in delay

The buffering requirements for SR class X consist of the following three components:

- d) A contribution from the queuing delay, equal to $maxBurst_X$
- e) The fan-in burst data size, computed as described in L.3.1.2
- f) The “permanent” buffer contents, which is equal to the fan-in burst data size.

The fan-in burst size and the permanent buffer contents do not count twice; if a fan-in burst happens, then it becomes the permanent buffer contents. If it happens a second time, it can only happen after a pause that has allowed the permanent buffer loading to drain. So the worst case buffering requirement for SR class X is simply the sum of the first two elements, $maxBurst_X$ [Equation (L.42)] and fan-in burst data size, so:

$$maxBuffering_X = maxBurst_X + (\text{fan-in burst data size})_X \quad (\text{L.45})$$

The worst-case total buffering requirement for all of the SR classes on a given Port is the sum of the following two parts:

- g) The worst-case buffering requirement for the lowest priority SR class on that Port
- h) One max-size frame $M_{i,Z}$ for each class Z of higher priority than Class X for each input port i.

Since the worst case buffer requirement for Class X is when it is taking almost all of the bandwidth from the higher-priority classes, the SR priority levels can share buffering space. The only additional requirement is for the fan-in from higher priorities on other Ports. Thus, the SR classes would do well to share their buffer space on each Port.

L.4 Operation of the credit-based shaper in a coordinated shared network

A Coordinated Shared Network (CSN) is a contention free, QoS capable, time division multiplexed access, network. One of the nodes of the network acts as the Network Coordinator (NC) node granting transmission opportunities to the other nodes of the network. The NC node also acts as the QoS manager of the network.

From the perspective of the specification of the credit-based shaper, a CSN is equivalent to a distributed bridge. Rather than shaping the AV traffic in the egress nodes, which could lack the buffering resources to do so, the shaping could instead be performed by the CSN's Network Coordinator per Class of Service when it schedules transmissions opportunities from the CSN's ingress to the CSN egress nodes. The behavior of CSN's egress to another MAC technology should be consistent with the behavior specified in this standard, regardless of how the shaper is implemented.

Annex M

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

- [B1] Alizadeh, M., Atikoglu, B., Kabbani, A., Lakshmikantha, A., Pan, R., Prabhakar, B., and Seaman, M., "Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization," *Proceedings of The 46th Annual Allerton Conference on Communication, Control and Computing*, Urbana-Champaign, Sept. 2008, Invited paper.
- [B2] 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>.)
- [B3] Binary Increase Congestion control—Transmission Control Protocol (BIC-TCP); <http://netsrv.csc.ncsu.edu/twiki/bin/view/Main/BIC>.
- [B4] Calculating the Delay Added by Qav Stream Queue (see <http://www.ieee802.org/1/files/public/docs2009/av-fuller-queue-delay-calculation-0809-v02.pdf>).
- [B5] Ethernet Local Management Interface (E-LMI), Metro Ethernet Forum Technical Specification MEF 16, 2006; <http://www.metroethernetforum.org/PDFs/Standards/MEF16.pdf>.
- NOTE—See Figure 5.3—ABTP common stream data header format. Also see IEC 61883-6, Consumer audio/video equipment—Digital Interface—Part 6: Audio and music data transmission protocol, Second edition, 2005-10, and IEC 61883-4, Consumer audio/video equipment—Digital Interface—Part 4: MPEG2-TS data transmission, Second edition, 2004-08.
- [B6] IEEE Std 802a-2003, IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture—Amendment 1: Ethertypes for Prototypes and Vendor-specific Protocol Development.⁵⁶
- [B7] IEEE Std 802.1AE-2006, IEEE Std for Media Access Control (MAC) Security.
- [B8] IEEE Std 802.1AS, IEEE Standard for Local and metropolitan area networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.
- [B9] IEEE Std 802.1D, 1993 Edition [ISO/IEC 10038:1993], IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local area networks—Media Access Control (MAC) bridges.
- [B10] IEEE Std 802.11-2007, Standard for LAN/MAN—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
- [B11] IEEE Std 1722™, IEEE Standard for Layer 2 Transport Protocol for Time Sensitive Applications in Bridged Local Area Networks.

⁵⁶IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854-4141, USA (<http://standards.ieee.org/>).

- [B12] IETF RFC 793 (STD0007), Transmission Control Protocol.⁵⁷
- [B13] IETF RFC 1323 (Proposed Standard), TCP Extensions for High Performance.
- [B14] IETF RFC 1633 (June 1994), *Integrated Services in the Internet Architecture: An Overview*, Braden, R., Clark, D., and Shenker, S.
- [B15] IETF RFC 2210 (Sept. 1997), *The Use of RSVP with IETF Integrated Services*, Wroclawski, J.
- [B16] IETF RFC 2211 (Sept. 1997), *Specification of the Controlled-Load Network Element Service*, Wroclawski, J.
- [B17] IETF RFC 2212 (Sept. 1997), *Specification of Guaranteed Quality of Service*, Schenker, S., Partridge, C., and Guerin, R.
- [B18] IETF RFC 2215 (Sept. 1997), *General Characterization Parameters for Integrated Service Network Elements*, Shenker, S., and Wroclawski, J.
- [B19] IETF RFC 2233 (Nov. 1997), *The Interfaces Group MIB using SMIv2*, McCloghrie, K., and Kastenholz, F.
- [B20] IETF RFC 2309 (Apr. 1998), *Recommendations on Queue Management and Congestion Avoidance in the Internet*, Braden, R., Clark, D., Crowcroft, J., Davie, B., Deering, S., Estrin, D., Floyd, S., Jacobson, V., Minshall, G., Partridge, C., Peterson, L., Ramakrishnan, K., Shenker, S., Wroclawski, J., and Zhang, L.
- [B21] IETF RFC 2475 (Dec. 1998), *An Architecture for Differentiated Services*, Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W.
- [B22] IETF RFC 2581 (Proposed Standard), TCP Congestion Control.
- [B23] IETF RFC 2597 (June 1999), *Assured Forwarding PHB Group*, Heinanen, J., Baker, F., Weiss, W., and Wroclawski, J.
- [B24] IETF RFC 2814 (May 2000), *SBM (Subnet Bandwidth Manager): A Protocol for Admission Control over IEEE 802-style Networks*, Yavatkar, R., Hoffman, D., Bernet, Y., Baker, F., and Speer, M.
- [B25] IETF RFC 2815 (May 2000), *Integrated Service Mappings on IEEE 802 Networks*, Seaman, M., Smith, A., Crawley, E., and Wroclawski, J.
- [B26] IETF RFC 2816 (May 2000), *A Framework for Providing Integrated Services Over Shared and Switched LAN Technologies*, Ghanwani, A., Pace, W., Srinivasan, V., Smith, A., and Seaman, M.
- [B27] IETF RFC 2960 (Proposed Standard), Stream Control Transmission Protocol.
- [B28] IETF RFC 3031 (Proposed standard), Multiprotocol Label Switching Architecture; The Internet Society; <http://www.ietf.org/rfc/rfc3031.txt>.
- [B29] IETF RFC 3246 (Mar. 2002), *An Expedited Forwarding PHB (Per-Hop Behavior)*, Davie, B., Charny, A., Baker, F., Bennet, J., Benson, K., Boudec, J., Chiu, A., Courtney, W., Davari, S., Firoiu, V., Kalmanek, C., Ramakrishnam, K., and Stiliadis, D.

⁵⁷Internet RFCs are retrievable by FTP at ds.internic.net/rfc/rfcnnnn.txt (where nnnn is a standard's publication number such as 1042), or call InterNIC at 1-800-444-4345 for information about receiving copies through the mail.

- [B30] IETF RFC 3270 (May 2002), *Multi-Protocol Label Switching (MPLS) Support of Differentiated Services*, Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and Heinanen, J.
- [B31] IETF RFC 4663 (Sept. 2006), *Transferring MIB Work from IETF Bridge MIB WG to IEEE 802.1*, D. Harrington.
- [B32] ISO/IEC TR 11802-1:1997, Information Technology—Telecommunications and Information Exchange Between Systems—Local and Metropolitan Area Networks—Technical Reports and Guidelines—Part 1. The Structure and Coding of Logical Link Control Addresses in Local Area Networks.⁵⁸
- [B33] ITU-T G.806 (2006), Characteristics of Transport Equipment—Description Methodology and Generic Functionality.⁵⁹
- [B34] ITU-T Recommendation G.8010/Y.1306 (02/2004), Architecture of Ethernet layer networks.
- [B35] ITU-T Recommendation G.8011/Y.1307 (08/2004), Ethernet over Transport—Ethernet services framework.
- [B36] ITU-T Recommendation G.8011.1/Y.1307.1 (08/2004), Ethernet private line service.
- [B37] ITU-T Recommendation G.8012/Y.1308 (08/2004), Ethernet UNI and Ethernet NNI.
- [B38] ITU-T Recommendation G.8021/Y.1341 (08/2004), Characteristics of Ethernet transport network equipment functional blocks.
- [B39] ITU-T Recommendation G.8031/Y.1342 (11/2009), Ethernet linear protection switching.
- [B40] ITU-T I.610 (02/1999), B-ISDN operation and maintenance principles and functions.
- [B41] ITU-T X.25 (10/1996), 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.
- [B42] MoCA MAC/PHY SPECIFICATION v1.0, MoCA-M/P-SPEC-V1.0-07122009, Multimedia over Coax Alliance (MoCA), July 12, 2009 (www.mocalliance.org).
- [B43] MoCA MAC/PHY SPECIFICATION EXTENSIONS v1.1, MoCA-M/P-SPEC-V1.1-06162009, Multimedia over Coax Alliance (MoCA), June 16, 2009 (www.mocalliance.org).
- [B44] MoCA MAC/PHY SPECIFICATION v2.0—Draft (MoCA-M/P-SPEC-V2.0-MMDDYYYY), Multimedia over Coax Alliance (www.mocalliance.org).
- [B45] 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/>).

⁵⁸ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iso.ch/>). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (<http://global.ihg.com/>). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

⁵⁹ITU-T publications are available from the International Telecommunications Union, Place des Nations, CH-1211, Geneva 20, Switzerland/Suisse (<http://www.itu.int>).

[B46] SMPTE 259M-2008, SMPTE Standard for Television—SDTV Digital Signal/Data—Serial Digital Interface, Society of Motion Picture and Television Engineers, 2008. See clause 8.

[B47] SMPTE 292M-2008, SMPTE Standard 1.5 Gb/s Signal/Data Serial Interface, Society of Motion Picture and Television Engineers, 2008. See subclause 4.3.

[B48] SMPTE 424M-2008, SMPTE Standard for Television—3 Gb/s Signal/Data Serial Interface, Society of Motion Picture and Television Engineers, 2008. See subclause 4.3.